Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Bor Juroš

Prepoznava pehotnih jarkov na podlagi višinskih map terena

diplomsko delo

univerzitetni interdisciplinarni študij
računalništvo in matematika

Mentor: doc. dr. Matej Kristan

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V nalogi naslovite problem avtomatske analize terena na podlagi slik posnetih z zraka s sistemom LIDAR. Natančneje, osredotočite se na problem detekcije pehotnih jarkov na razgibanem terenu. Preglejte sorodna dela na področju zaznavanja jarkom podobnih struktur v slikah, ter izberite nabor najbolj obetavnih metod za reševanje problema detekcije na vaši domeni. Metode opišite ter predlagajte lasten algoritem za detekcijo jarkov. Uspešnost algoritma analizirajte na zbirki realnih posnetkov terena z jarki ter izpostavite prednosti in slabosti.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Bor Juroš sem avtor diplomskega dela z naslovom:

Prepoznava pehotnih jarkov na podlagi višinskih map terena

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mateja Kristana.
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 18. septembra 2015 Podpis avtorja:

Zahvaljujem se svojemu mentorju doc. dr. Mateju Kristanu za vso strokovno pomoć, potrpežljivost in spodbudo pri nastajanju diplomskega dela.
Posebna zahvala gre doc. dr. Primož Banovcu, ki je bil idejni vodja diplomske naloge in mi je omogočil dostop do podatkov, uporabljenih v diplomskem delu. Prav tako se iskreno zahvaljujem njemu ter vsem ostalim zaposlenim na Inštitutu za vodarstvo za vso pomoč in spodbudo pri izdelavi diplomske naloge.
Zahvaljujem se vsem sodelavcem iz podjetja XLAB d.o.o. za razumevanje in podporo, ki sem je bil deležen med študijem.
Prav tako se zahvaljujem Petri, za vso pomoč ter dejstvo, da me je uspešno prenašala, ko je bilo to najtežje.
Nenazdnje pa se iskreno zahvaljujem družinskim članom, staršem ter bratcu Filipu, ki so mi vedno stali ob strani.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Sorodna dela 2

1.2 Cilji in prispevki 4

1.3 Zgradba 4

2 Predstavitev metod in algoritmov **5**

2.1 Sistem Lidar 5

2.2 Predstavitev in priprava podatkov 8

2.3 Konvolucija s filtri za glajenje 12

2.4 Dinamično nastavljanje pragovne vrednosti - trikotniška metoda 15

2.5 Detekcija maksimalno stabilnih regij 19

2.6 Lokalni binarni vzorci 22

2.7 Sobelov filter za detekcijo robov 24

3 Naš pristop k detekciji jarkov **27**

3.1 Glajenje terena 27

3.2 Priprava terenskih značilnic 29

3.3 Končna obdelava značilnic 35

4 Eksperimentalna analiza **41**

4.1 Parametri 41

4.2 Testni podatki in testiranje 45
4.3 Predstavitev rezultatov 50
4.4 Analiza pridobljenih rezultatov in primerjava z izhodiščno metodo 51
4.5 Kritični primeri 55

5 Zaključek **59**
5.1 Ugotovitve 59
5.2 Uporaba in nadaljnje delo 61

Seznam uporabljenih kratic

kratica	angleško	slovensko
LIDAR	light detection and ranging	svetlobno zaznavanje in merjenje oddaljenosti
MSER	maximally stable extremal regions	maksimalno stabilne regije
SVM	support vector machine	metoda podpornih vektorjev
GPS	global positioning system	globalni sistem pozicioniranja
SLAM	simultaneous localization and mapping	hkratna lokalizacija in mapiranje

Povzetek

Ostanki pehotnih jarkov igrajo pomembno vlogo pri razumevanju in rekonstrukciji dogajanj v najtemnejših letih moderne zgodovine. Omogočajo nam vpogled v dogajanje in potek prve svetovne vojne tako na naših tleh, kakor trudi drugod po Evropi. Z razvojem sistema Lidar in napredkom na področjih mapiranja in topologije, imamo sedaj možnost vpogleda v prej nedostopna območja. V delu tako prestavimo sistem za avtomatsko detekcijo pehotnih jarkov na razgibanih terenih. To stori na podlagi višinskih map terena pripravljenih iz točkovnega oblaka, ki ga ustvari sistem Lidar. Na višinskih mapah so nato uporabljeni algoritmi s področja računalniškega vida in drugih sorodnih območji, s pomočjo katerih algoritem detektira in označi pehotne jarke. Vidimo lahko da algoritem prekaša referenčno metodo, hkrati pa dopušča veliko maneverskega prostora za nadgradnje in nadaljni razvoj.

Ključne besede: Lidar , GPS, računalniški vid, MSER, LBP, konvolucija .

Abstract

Remains of trenches used by infantry during World War 1 play an important role in understanding and reconstructing the happenings during one of the darkest eras of human history. They allow us to look at and simulate the course of war in Europe. With developements of Lidar systems and progress in the areas of mapping and topology, we can now observe the terrain previously hidden from our view. In our thesis we present an algorithm for automatic detection of trenches on non-uniform terrain. The proposed algorithm uses principles of computer vision and applies them to the height map of the terrain, made from point cloud created by Lidar, to detect and mark trench-like structures. Results show that our approach outperforms a related method, while leaving enough room for possible upgrades and future work.

Keywords: Lidar, GPS, computer vison, MSER, LBP, convolution.

Poglavje 1

Uvod

V zadnjih desetletjih se na področju kartografije in raziskovanja terena vedno bolj uveljavlja sistem Lidar (angl. Light Detection And Ranging) [8]. Gre za meritveni sistem, ki spada v razširjeno družino radarjev, vendar Lidar za detekcijo razdalje ne uporablja radijski valov temveč laserske žarke. Podrobneje je delovanje sistema Lidar opisano v Poglavju 2.1.
Ker gre za skoncentrirane meritve (snop svetlobe vrne meritev razdalje v eni sami točki, saj se ne razprši), je sistem sposoben detekcije še tako majhnih nepravilnosti in lastnosti terena, če je le gostota meritev glede na površino dovolj visoka. Prav tako nam sistem Lidar omogoča vpogled na območja, ki so prekrita z rastlinjem, saj so svetlobni žarki dovolj močni, da prodrejo do tal, tako da lahko opazujemo lastnosti terena, ki nam je bil prvotno nedostopen ali pa bi bilo natančno preiskovanje na terenu logistično preveč zahtevno. Ena od terenskih nepravilnosti, ki so jih pri nas gozdovi dodobra zakrili, so tudi pehotni jarki, ki so ostali iz časa prve svetovne vojne [5].
Ko iz podatkov pridobljenih s sistemom Lidar ustvarimo 3D model terena, hitro postane jasno, da je na naših tleh ostalo mnogo več jarkov, kolikor jih je označenih in prepoznanih v zgodovinskih arhivih. Detekcija in označevanje jarkov, bi tako bila v veliko pomoč zgodovinarjem, saj bi izboljšan pregled jarkov omogočal boljši vpogled v dogajanje v prvi svetovni vojni. Vendar pa gre pri avtomatski detekciji jarkov za dokaj težak problem; vizualno je namreč

jarke prepoznati dokaj preprosto, vendar gre za izjemno neenakomerne struktrure, ki se spreminjajo glede na svojo okolico. CIlj naše diplomske naloge je tako bila izdelava algoritma, ki bi jarke identificiral ne glede na prej omenjene okoliščine. Za oporo so nam tako služili članki in raziskave, z različnih področji vizualnega zaznavanja, vizualizacije v medicini, proučevanja površja, ki so predstavljeni v Poglavju 1.1.

1.1 Sorodna dela

Z razvojem sistema Lidar in posledično s cenovno dostopnostjo le-tega, se je njegova uporaba razširila na različna področja. Poleg raziskav v smeri detekekcije splošnih značilnosti okolja na podlagi podatkov pridobljenih s sistemom Lidar [16], ki bi propomogle k reševanju problema hkratne lokalizacije in mapiranja(angl., simultaneous localization and mapping, SLAM) problemov v robotiki [35], je bilo več raziskav posvečenih detekciji prelomov v ledenih ploskvah [28, 9, 25], kjer so za detekcijo prelomnic merili razpršenost odbitega žarka, saj se le ta razprši drugače, glede na lastnosti materijala, ki ga zadane. V primeru detekcije ledenih ploskev in razpok med njimi, je na razpršenost vplivala temperatura vode, ki je bila v razpokah višja kakor v okolici; več o lastnostih sistemov Lidar, ki take meritve omogočajo, je napisanega v Poglavju 2.1.

Podobne raziskave so bile opravljene na tematiko avtomatske detekcije cest, na podlagi visoko ločljivostnih satelitskih slik [20, 15], kjer so se soočali s podobnimi ovirami, kot pri detekciji jarkov: sence, šum, delno zakriti objekt, itd. Izbrana sta bila dva pristopa, pri prvem [20], so avtomatizirali prepoznavo cest z uporabo strojnega učenja na podlagi nevronskih mrež, pri drugem [15], pa so se problema detekcije lotili na podlagi topoloških značilnosti in ekstrakcije robov.

Sorodni, vendar bolj kompleksni algoritmi za avtomatsko detekcijo, se s tehnološkim napredkom pojavljajo tudi v zdravstvu, kjer je ena od tem detekcija drevesnih struktur v 2D in 3D posnetkih. Predvsem gre za algoritme, ki de-

tektirajo in označujejo nevrone [30, 32, 29]. Algoritmi predprocesirajo slike na podlagi gradientov [29], nato pa v slikah iščejo linearne strukture, katere nato na podlagi statističnih modelov povežejo v drevesa [30, 32]. Prednost takih algoritmov je, da se zelo dobro obnesejo, tudi ko jih aplicirajo na nemedicinske podatkovne sete, kjer želimo detektirati povezane linearne strukture. Algoritem za detekcijo nevronov [32] tako omogoča izjemno dobro prepoznavo cest na podlagi posnetkov iz zraka. Takšni algoritmi za dosego končnih rezultatov najpogosteje uporabijo različne statistične modele na podlagih katerih iz detektiranih območji z različnimi stopnjami verjetnosti ustvarijo drevesno oz. povezano strukturo [34, 32].

Slika 1.1: Primer uporabe algoritma za detekcijo nevronov [32], za detekcijo cest. Drevesna struktura je namerno zamaknjena, za lažjo vizualizacijo. Slika vzeta iz [32].

1.2 Cilji in prispevki

Cilj diplomske naloge je bil razvoitialgoritem za avtomatsko detekcjo pehotnih jarkov iz prve svetovne vojne na podlagi podatkov pridobljenih s sistemom Lidar. Želeli smo doseči vsaj 75% uspešnosti pri prepoznavi jarkov, s kar najmanjšo stopnjo nepravilno pozitivno prepoznanih območji, ki bi jih nato na podlagi verjetnosti povezali v omrežja. Poleg detekcije jarkov smo želeli raziskati primernost takega algoritma za detekcijo vrtač.
Uspešno smo izdelali in implementirali algoritem, ki za vhodni podatek prejme višinsko karto terena, kjer vsaka celica vsebuje koordinati, ki določata njeno lokacijo v svetu ter nadmorsko višino, kot izhodni podatek pa algoritem vrne binarno masko z označenimi jarki, ki še vedno vsebuje koordinate, kar omogoča njeno uporabo v računalniških programih za obdelavo topoloških podatkov. Razen na kritičnih območjih predstavljenih v Poglavju 4.5, kjer je učinkovitost algoritma padla, smo želeni odstotek prepoznave dosegli. Prav tako smo uspešno izdelali primer algoritma za detekcijo vrtač, ki temelji na algoritmu detekcije maksimalno stabilnih regij (angl. maximally stable extremal regions, MSER).
V nadaljnem razvoju algoritma v sodelovanju z Inštitutom za vodarstvo bomo jarke povezali v omrežja, nato pa bomo na njihovi podlagi izdelali vektorski sloj, ki bo jarke označeval, ter ga ponudili v uporabo v Geografskih Informacijskih Sistemih (GIS).

1.3 Zgradba

V nadaljevanju je diplomska naloga razdeljena v štiri poglavja. Poglavje 2 vsebuje teoretično predstavitev algoritmov in metod, ki smo jih uporabili pri izdelavi naše diplomske naloge. V Poglavju 3 je po korakih predstavljem naš pristop k izdelavi diplomske naloge. Sledi vrednotenje uspešnosti našega algoritma v Poglavju 4, v zadnjem Poglavju (5) pa opišemo naše ugotovitve ter cilje, ki jih želimo doseči z nadaljnim razvijanjem našega algoritma.

Poglavje 2

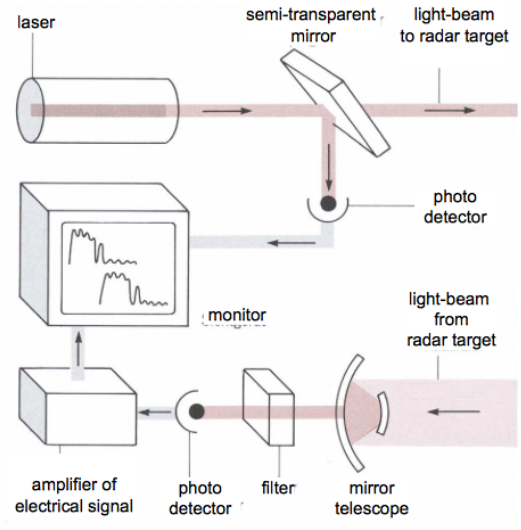
Predstavitev metod in algoritmov

2.1 Sistem Lidar

Začetki sistema Lidar (angl. light detection and ranging) segajo v 60. leta, ko je raziskave na področju svetlobne detekcije prevzela NASA (*National Aeronautics and Space Administration* - ameriška Nacionalna zrakoplovna in vesoljska uprava) [8]. Prvotni cilj ni bila izdelava topoloških map, temveč detekcije lastnosti atmosfere, vode, ledu, saj so želeli sisteme namestiti na vesoljska plovila, s katerimi bi nato raziskali vesolje. Hitro se je pokazalo, da so sistemi Lidar izjemno natančni, kar je v kombinaciji s sistemom GPS omogočalo, da so poleg vseh zahtevanih meritev lahko izrisali še izjemno natančno topološko mapo terena [26, 27].
Prvi komercialni sistemi Lidar so se nato pojavili šele v 90ih letih, predvsem zaradi cene ter dostopnosti zanesljivih sprejemnikov GPS [4]. Po letu 2000 je razširjenost takih in drugačnih sistemov, ki za detekcijo uporabljajo laserske žarke, strmo narasla, saj so se pojavili tudi prvi računalniški programi, ki so bili sposobni učinkovito obdelati tako enormne količine podatkov.
Poleg same vizualizacije terena, so podatki, ki jih priskrbi topološki Lidar izjemno uporabni tudi v drugih strokah. Največ jih tako uporabljajo pripadniki

strok, ki se ukvarjajo z opazovanjem in analiziranjem dogodkov v naravi. Lahko se jih uporabi za natančne izračune postavitve vodnih pregrad in jezov (pri nas se s tem ukvarja Inštitut za vodarstvo), za predvidevanje in simuliranje naravnih nesreč [14] (npr. vizualizacijo splazitve glede na spreminjanje snežne mase skozi čas, novo nastalih razpok [24]), iskanje različnih terenskih značilnosti (npr. nevarne razpoke v ledenih ploskvah) [28, 9], proučevanje vpliva človeka na okolje [33], itd.

Kot pove ime samo, sistem Lidar za detekcijo razdalje ter ostalih parametrov tarče uporablja laserske žarke, delovanje pa je prikazano na Sliki 2.1, kjer je prikazana poenostavljena zgradba sistema Lidar. Laserski žarek se s pomočjo pol-prosojnega ogledala razcepi, eden od žarkov, je takoj usmerjen v fotodetektor in bo služil za prepoznavo in čas odboja, drugi žarek pa potuje proti tarči. Odbiti žarek je nato s pomočjo leče detektiran in usmerjen v sprejemni fotodetektor ter od tam naprej v procesno enoto. S primerjavo obeh žarkov ter časom, ki je potekel do zaznave drugega žarka, se po enačbi za sistem Lidar lahko izračuna oddaljenost tarče ter ostale parametre, ki so odvisni od različice sistema Lidar ter tipa žarka, ki je v uporabi.



Slika 2.1: Poenostavljena zgradba sistema Lidar. Slika vzeta iz [3].

Fizikalno ozadje sistema Lidar

V tem poglavju bomo predstavili enačbo, ki opisuje relacijo med številom fotonov v oddanem žarku in številom fotonov v zaznanem žarku, glede na pogoje v katerih sitem Lidar uporabljamo. Enačba tako upošteva sipanje svetlobnega žarka v atmosferi, interakcijo med žarkom in objekti, zgradbo sistema Lidar, ki je v uporabi ter tip žarka oz. valovno dolžino žarka, ki je bil oddan. V uporabi sta dva ekvivalentna zapisa enačbe, prvi opisuje relacijo v odvisnosti od oddanih ter prejetih fotonov, drugi pa v odvisnosti od oddane in prejete moči svetlobnega žarka. Najpogosteje se enačba tako zapiše v spodnji obliki:

$$p(r) = \frac{C\beta(r)t^2(r)}{r^2}, \tag{2.1}$$

kjer $p(r)$ predstavlja prejeto oz. izmerjeno moč žarka na razdalji r, β predstavlja koeficient povratnega sipanja v prostoru na na razdalji r, C je konstanta odvisna od sistemskih parametrov sistema Lidar: moč oddanega žarka, dolžina pulza ter lastnosti prejemnika žarkov. $t^2(r)$ predstavlja dvosmerno propustnost na razdalji r, glede na medij skozi katerega žarek potuje, v našem primeru je to največkrat atmosfera. $t^2(r)$ je preko enačbe (2.2) povezan s koeficientom izgube γ, odvisnga od medija skozi katerega žarek potuje, ki zajema absorbcijo in sipanje žarka med potovanjem skozi medij.

$$t^2(r) = \exp\left[-2 * \int_0^r \gamma(s)ds\right]. \tag{2.2}$$

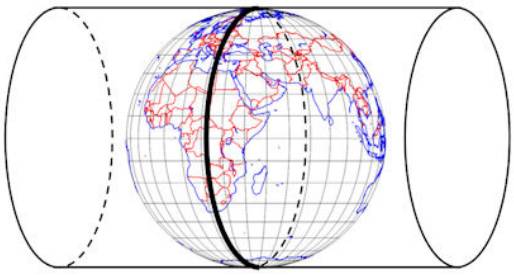
Kot smo omenili, sistemi Lidar za detekcijo uporabljajo različne tipe svetlobe, oz. različne valovne dolžine, glede na želeni rezultat. Topografski sistemi Lidar, katerih topografske mape smo uporabili v diplomski nalogi, navadno uporabljajo valovne dolžine okoli 1000 nm, saj je taka vrsta laserskih sistemov najbolj dostopna, cenovno ugodna in zagotavlja dovolj točne rezultate. Take vrste žarek pa ima seveda tudi svoje slabosti. Za varno delo v laboratorijih oz. takrat, ko obstaja nevarnost poškodb oči, je priporočljivo povečati valovno dolžino in s tem prepreči poškodbe oči. Ena od pomembnejših pomanjkljivosti standarnega žarka, je tudi ta, da se absorbira, ko zadane vodo,

zato je v primerih, ko želimo topološko karto dna vodnega telesa, potrebno valovno dolžino zmanjšati na razdaljo okoli 500nm in manj, da se prepreči izgube energije, ko žarek potuje skozi vodo.
Dodatno se lahko Lidar sisteme prilagodi tudi za identifikacijo trdnih materialov, kjer sprejemnik meri spektre odbite svetlobe [18]. Z uporabo Dopplerjevega efekta, pa lahko sistem Lidar uporabimo tudi za izračun hitrosti objekta [21].

2.2 Predstavitev in priprava podatkov

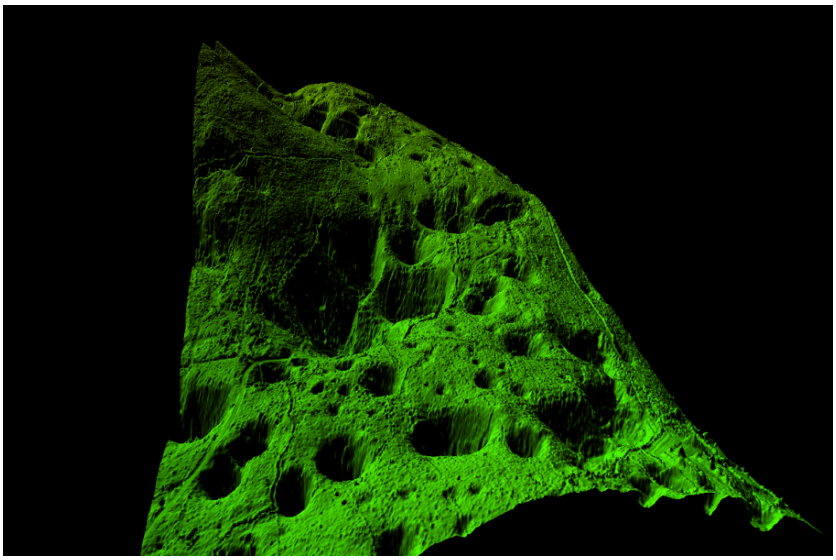
Podatki, ki jih sistem Lidar vrne po obdelavi, so predstavljeni kot točkovni oblak, kjer ima vsaka točka tri parametre x, y, z. x in y predstavljata zemljepisno širino in dolžino, z pa nadmorsko višino. V taki obliki obliki podatki niso primerni za uporabo v našem algoritmu, zato jih je bilo najprej potrebno obdelati in iz njih pripraviti višinske mape terena. To smo storili s pomočjo sistema Manifold, ki je namenjen prikazovanju in obdelavi topoloških podatkov. Glede na to, da so koordinate točk predstavljene v obliki zemljepisne širine in dolžine, jih je bilo najprej potrebno pretvoriti v državni koordinatni sistem. Za pretvorbo iz zemljepisnih v standardne koordinate je na voljo množica različnih projekcij, ki so odvisne od položaja zemlje, v našem primeru smo tako uporabili projekcijo Gauss-Kruger [12], ki Zemljo kot elipsoid projecira na navidezni cilinder, kateri se nato odpre v ravnino in zagotavlja točne rezultate za območja v pasu $\pm 15°$ od centralnega poldnevnika. Razlog za to je viden na Sliki 2.2. Zemlja se namreč projekcijskega cilindra dotika s centralnim poldnevnikom, tako da se pravilna razmerja pri projekciji ohranijo le v njegovi okolici. Pri bolj oddaljenih točkah pa pride do popačitve razdalj ter razmerji.

Tako projecirane točke je bilo nato moč vizualizirati v sistemu Manifold, ki iz točkovnega oblaka ustvari 3D model terena. Z rastlinjem odstranjenim v predpripravi podatkov (rastlinje je bilo odstranjeno še preden smo imeli



Slika 2.2: Demonstracija rotirane Mercatorjeve oz. Gauss-Kruger projekcije. Zemlja se kot elipsoid projecira na cilinder, katerega se nato odpre v ravnino. Lahko vidimo, da pravilna razmerja ohranijo samo točke blizu centralnega poldnevnika, ki se stika z valjem, zato taka projekcija zagotavlja natančnost samo na območjih $\pm 15°$ od centralnega poldnevnika. Slika vzeta iz [1].

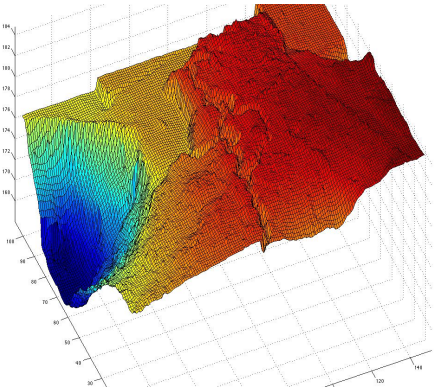
podatke na voljo), so bili jarki že vidni.



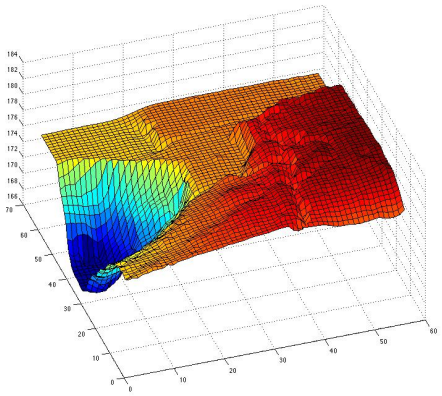
Slika 2.3: Jarki in vrtače vidni na 3D modelu površja v programu Manifold.

Še vedno pa je bilo iz točkovnega oblaka potrebno pripraviti višinske mape terena, ki bi jih uporabili v našem algoritmu. Tu je bilo pomembno, kako veliko celico si bomo izbrali v višinski mapi, če bo celica prevelika, bomo za določitev nadmorske višine v celici uporabili preveč točk, kar bo pripeljalo do tega, da se bodo mikro-topološke značilnosti, kot so jarki izgubile, če pa bodo

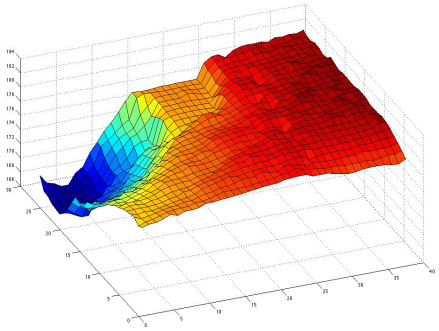
celice premajhne, pa bo prišlo do prevelike količine šuma. Z gostoto točk, ki je v povprečju znašala 5 točk na kvadratni meter, se je v praksi najbolje izkazala celica velikosti $1m \times 1m$. Na Sliki 2.4 lahko vidimo, kako velikost celice višinske mape vpliva na izgubo mikro-topoloških značilnosti. Predstavljene so višinske mape istega območja, z velikostjo celic $1m \times 1m$,$2m \times 2m$ ter $3m \times 3m$. Večje kot je območje celice, več točk bo vplivalo na določanje skupne višine za posamezno celico, s tem pa izgubimo mikro-topološke značilnosti terena.



(a) $1m \times 1m$.



(b) $2m \times 2m$.



(c) $3m \times 3m$.

Slika 2.4: Višinska mapa predstavljena v programu Matlab, z vidnim pehotnim jarkom. Izguba mikro-topoloških značilnosti s povečevanjem velikosti celice je očitna.

Poleg velikosti celice, smo določili tudi velikost območja, ki ga ena višinska mapa pokriva. Izbira velikosti višinske mape nima tako velikega vpliva na rezultate našega algoritma kot velikost celice, vseeno pa je bilo potrebno najti kompromis med velikostjo višinske mape - večja kot je mapa, počasnejše bodo operacije, ki jih nad njo izvaja algoritem, ter količino višinskih map - manjša kot bo višinska mapa, več jih bo potrebno pripraviti, da z njimi pokrijemo celotno območje. Na podlagi tega je bila izbrana višinska mapa velikosti $1km \times 1km$.

2.3 Konvolucija s filtri za glajenje

Eden od principov, ki smo jih uporabili pri izdelavi našega algoritma, je bila uporaba konvolucije. Gre za enega temeljnih postopkov pri procesiranju slik, ki nam omogoča, da sliko modificiramo tako, da izpostavimo oz. zakrijemo lastnosti, ki se na sliki pojavljajo. V primeru našega algoritma je šlo za konvolucijo v dveh dimenzijah, saj smo jo izvajali nad višinsko mapo terena, ki je bila v našem primeru predstavljena kot dvodimenzionalna matrika nadmorskih višin.
Sam proces konvolucije sestavljata dva elementa, slika S, nad katero bomo konvolucijo izvedli, ter jedro k, ki je navadno manjša matrika od S, s fiksnimi vrednostmi.

$$S = \begin{bmatrix} 1 & 3 & 5 & 2 & 4 \\ 2 & 8 & 7 & 3 & 9 \\ 1 & 1 & 8 & 4 & 7 \\ 2 & 4 & 6 & 1 & 3 \\ 7 & 2 & 9 & 8 & 1 \end{bmatrix}, \quad k = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

V tem primeru smo za k izbrali Sobelov [31] operator, ki poudari robove v horizontalni smeri. Nato po sledeči enačbi izračunamo njuno konvolucijo:

$$c(n_1, n_1) = \sum_{k_1=0}^{r-1} \sum_{k_2=0}^{c-1} S(k_1, k_2) k(n_1 - k_1, n_2 - k_2), \tag{2.3}$$

kjer $c(n_1, n_2)$ predstavlja vrednost konvolucije v polju (n_1, n_2), r in c pa predstavljata število stolpcev ter vrstic v S. Kot rezultat konvolucije dobimo naslednjo matriko:

$$S * k = \begin{bmatrix} 12 & 25 & 25 & 22 & 21 \\ -2 & -1 & 6 & 10 & 8 \\ -4 & -9 & -8 & -11 & -14 \\ 13 & 9 & 7 & 3 & -8 \\ -8 & -16 & -17 & -11 & -7 \end{bmatrix}.$$

Pri izdelavi algoritma smo tako uporabil tri različne filtre: uniformni filter, Gaussov filter ter ciklični uniformni filter.

Uniformni filter: U, v katerem so vse vrednosti jedra enake $k(n_1, n_2) = \frac{1}{r \times c}$, r in c pa sta velikosti vrstic oz. stolpcev, tako da v vsaki celici kot rezultat konvolucije dobimo povprečje vseh sosedov, ki smo ga uporabili za glajenje matrike.

$$U = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Gaussov filter: G, ki je diskretna aproksimacija Gaussove porazdelitve v dveh dimenzijah, definirane z enačbo:

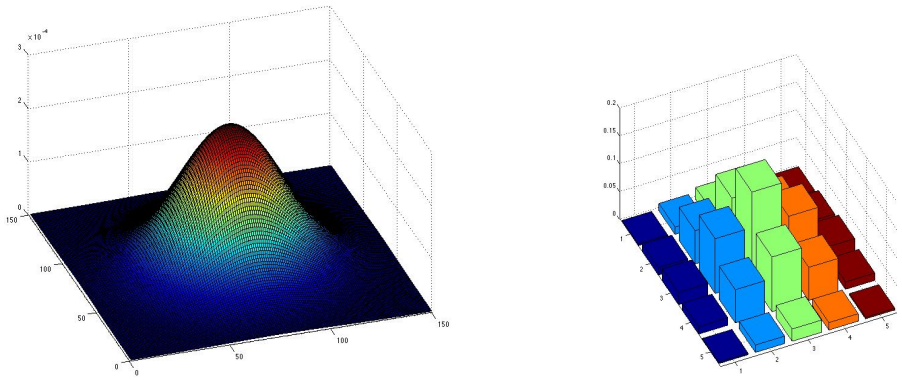
$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right] \tag{2.4}$$

in se uporablja za zameglitev slik. Primera zveznega Gaussovega jedra ter njegove aproksimacije z matriko velikosti 5×5 sta prikazana na Sliki 2.5. Vidimo, da manjša kot je matrika s katero aproksimiramo Gaussovo jedro, bolj groba bo aproksimacija.

Gaussov filter nam tako lahko vrne zelo podobne rezultate, kakor uniformni filter, vendar lahko pri Gaussovem jedru kontroliramo standardno deviacijo σ, kar bo vplivalo na rezultat. Večja kot je standardna deviacija, bolj položna je Gaussova krivulja, ki jo jedro aproksimira in posledično je Gaussov filter

vedno bolj podoben unfiormnemu. Spodaj je predstavljen primer 5×5 Gaussovega filtra s standardno deviacijo $\sigma = 1$.

$$G = \frac{1}{273} * \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$



(a) Zvezno gaussovo jedro. (b) Diskretna aproksimacija.

Slika 2.5: Predstavitev zvezne Gaussove funkcije dveh spremenljivk ter njena diskretna aproksimacija s filtrom velikosti 5×5.

Ciklični uniformni filter: C, ki je zelo podoben standardnemu uniformnemu filtru, od njega se razlikuje po obliki, saj ni pravokotne temveč ciklične oblike ter po tem, da je hkrati tudi utežen, tako da imajo na rezultat konvolucije večji vpliv celice, ki so bližje centru filtra. Spodaj je predstavljen primer 3×3 cikličnega uniformnega filtra.

$$C = \begin{bmatrix} 0.025 & 0.145 & 0.025 \\ 0.145 & 0.32 & 0.145 \\ 0.025 & 0.145 & 0.025 \end{bmatrix} \tag{2.5}$$

2.4 Dinamično nastavljanje pragovne vrednosti - trikotniška metoda

Nastavljanje pragovne vrednosti oz. upragovanje je postopek, pri katerem za sliko (oz. množico podatkov) S določimo prag p, navadno tako da $min(S) \leq p \leq max(S)$, saj nastavljanje pragovne vrednosti izven tega intervala nima smisla. Vpliv pragovne vrednosti je viden na Sliki 2.6, kjer so predstavljeni rezultati upragovanja z različnimi pragovnimi vrednostmi, ki tečejo od najmanjše do največje vrednosti v sliki.

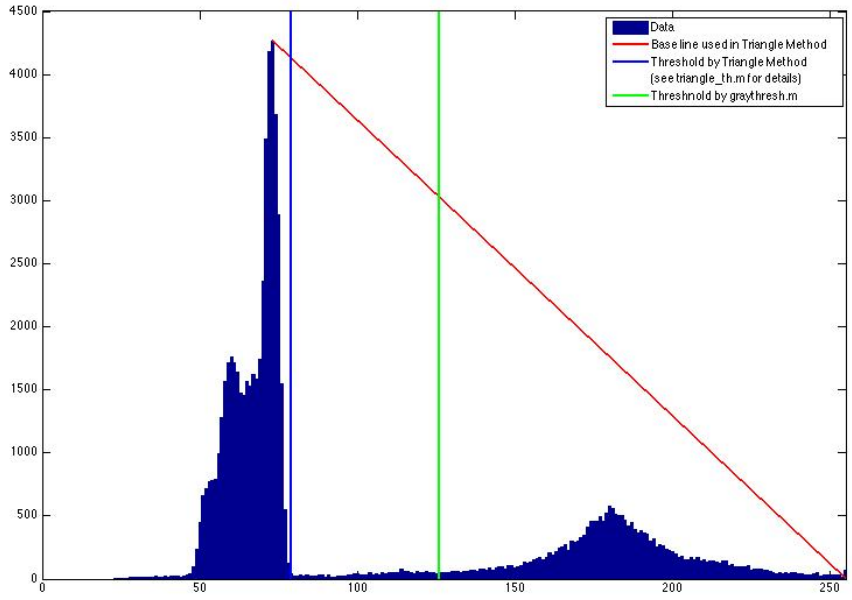
Če funkcijo upragovanja označimo z $o(x)$, potem lahko upragovanje opišemo na sledeči način:

$$\forall x \in S \Rightarrow o(x) = \begin{cases} 1 & ; x \geq p \\ 0 & ; x < p \end{cases}. \tag{2.6}$$

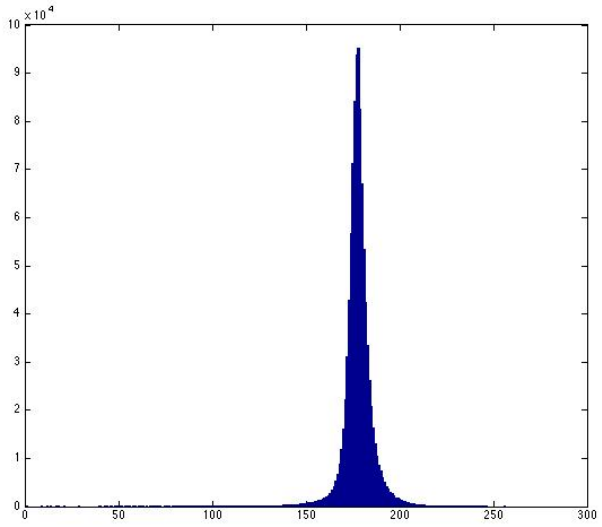
Tako iz običjane dobimo binarno sliko. Problem dinamičnega nastavljanje pragovne vrednosti se pojavi takrat, ko želimo upragovati slike/matrike, kjer se pragovne vrednosti razlikujejo od matrike do matrike. Iskali smo algoritem za dinamično nastavljanje pragovne vrednosti, ki bo poiskal kar najboljši prag za tipe matrik, s katerimi smo operiral. Izkazalo se je, da je za naše potrebe najboljši pristop dinamično upragovanje po trikotniški metodi [13], ki je primerna za dokaj ozek nabor matrik oz. njihovih histogramov. Gre za matrike katerih histogrami imajo točno določeno obliko, predstavljeno na Sliki 2.7. Taki histogrami imajo zelo izrazit vrh na eni strani ter nekaj manjših vrhov drugje. Algoritem nato poveže najvišji vrh z najbolj oddaljeno točko na histogramu in za vsako ne-ničelno točko histograma izračuna razdaljo med vrednostjo v tej točki do povezovalne črte. Algoritem prag postavi tam, kjer je razdalja največja, saj se na tem mestu histogram naj-

Slika 2.6: upragovanje po korakih, višji kot je prag, manj je pozitivnih regij. V zadnjem koraku pragovna vrednost zavzame maksimalno vrednost v sliki, pozitivnih regij ni.

bolj strmo povzpne. S tem določimo prag tako, da je najbolj izrazita grupa ločena od vseh ostalih. Na Sliki 2.8 lahko vidimo obliko histograma, ki je bila značilna za vse naše višinske mape. Vidimo lahko izrazito gručo vrednosti, ki predstavlja območja, ki nas ne zanimajo. Levo od nje pa se nahajajo mnogo manjše gruče vrednosti, ki predstavljajo jarke, vrteče in ostale mikrotopološke značilnosti.
Naknadno smo se ukvarjali še z dodatno metodo avtomatskega nastavljanja pragov, ki temelji na odvodih [17]. Ideja je, da se določi vse vrhove v histogramu, nato pa se jim aproksimira krivuljo. Glede na to da poznamo obliko histograma, lahko nato iz vrednosti odvodov izračunamo, na katerem mestu je ukrivljenost krivulje največja. To bo ponovno ravno tam, kjer se histogram najbolj strmo vzpne, tako da dobimo zelo podoben prag kakor po trikotniški metodi.



Slika 2.7: Primer delovanja algoritma za določanje pragovne vrednosti na podlagi trikotniške metode. Z zeleno je označen prag, ki ga določi osnovna metoda upragovanja v programu Matlab, z modro pa je označen prag izbran s trikotniško metodo.

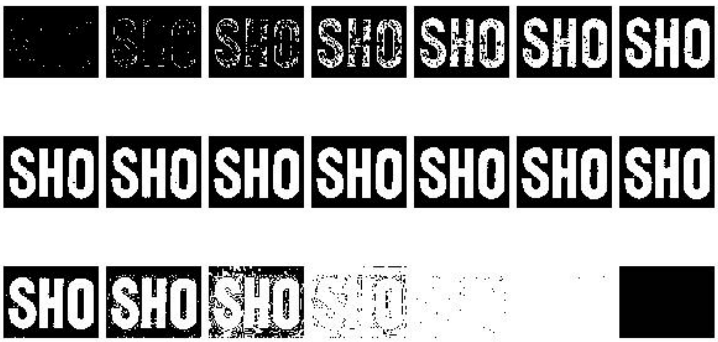


Slika 2.8: Standardna oblika histograma za naše višinske mape.

2.5 Detekcija maksimalno stabilnih regij

Detekcija maksimalno stabilnih regij (angl. maximaly stable extremal regions, SLAM), je pristop za detekcijo območij v sliki (ang. *Blob detection*), ki se od okolice razlikujejo po enem od atributov npr. barva, svetlost, Ideja algoritma MSER je, da izberemo regije, ki se pri upragovanju z različnimi pragovnimi vrednostmi čim manj spreminjajo. Algoritem si lahko predstavljamo po korakih:

1. Za izbrano sliko S izvedemo serijo upragovanj, s pragom t, ki na začetku zavzame najmanjšo vrednost v naši sliki, nato pa se pragovna vrednost povečuje za korak Δ, dokler ne doseže maksimalne vrednosti v S.



Slika 2.9: Sliko S upragujemo glede na pragovno vrednost t, tako dobimo serijo binarnih slik.

2. Skozi serijo upragovanih slik poiščemo povezana območja. Na Sliki 2.9 lahko vidimo, kako na drugi zaporedni sliki postanejo vidna prva pozitivna območja. Vsaka skupina povezanih celic postane kandidat za maksimalno stabilno regijo in se od svojega prvega pojava spremlja skozi vse nadaljne korake upragovanja..

3. Za vsako od povezanih območji, detektiranih v prejšnjem koraku, je sedaj potrebno ugotoviti kdaj je maksimalno stabilno, to storimo tako,

da poiščemo lokalni minimum sledeče funkcije:

$$q(x) = \frac{|Q_{i+\Delta} - Q_{i-\Delta}|}{Q_i}. \tag{2.7}$$

S Q_i je označeno št. celic oz. površina regije pri i-tem koraku upragovanja, z Δ pa je označen korak upragovanja. Funkcija $q(x)$ tako računa prirastek novih celic k posamezni regiji med korakoma $i - \Delta$ ter $i + \Delta$, ki je nato normaliziran s številom celic v koraku i, saj se povezana območja močno razlikujejo po velikosti.

4. Rezultat algoritma MSER je identificiranje regij, ki so sedaj maksimalno stabilne ter za katere velja, da je vrednost vseh celic, ki so v območju vsebovana večja oz. manjša od vrednosti, na katere območje meji, zato se v imenu algoritma pojavi ime *ekstremalna* regija.

5. V zadnjem koraku je potrebno izločiti tista območja, ki jih je algoritem sicer prepoznal kot maksimalno stabilna, a ne zadostujejo podanim parametrom. Osnovna kriterija sta minimalna ter maksimalna velikost prepoznanih območij. Okvirno vemo, kakšne regije lahko pričakujemo, ko algoritem poženemo, zato lahko interval velikosti tudi primerno nastavimo, vse regije, ki so tako po površini oz. številu celic prevelike oz. premajhne nato algoritem izloči. Dodatni parameter, ki vpliva na to koliko različnih regij nam bo algoritem označil, pa je maksimalna variacija območja Γ_{max}. Pri pogoju maksimalne dopuščene variacije ponovno uporabimo funkcijo:

$$q(x) = \frac{|Q_{i+\Delta} - Q_{i-\Delta}|}{Q_i}, \tag{2.8}$$

katere lokalni minimum smo iskali za določitev maksimalno stabilne regije. Čeprav smo za funkcijo variacije poiskali lokalni minimum, pa je lahko variacija območja še vedno zelo visoka. S parametrom maksimalne variacije tako lahko določimo, koliko raznolike regije še dopuščamo. Za zelo izrazite regije je variacija nizka, tako da če povečujemo maksimalno variacijo (tipično zavzame vrednosti med 0.1 in 1), nam

bo algoritem vrnil več regij, a ni nujno da bodo tako stabilne. Na Sliki 2.10 je tako prikazan rezultat, ki ga vrne algoritem MSER. Vidimo da so vse tri črke označene kot prepoznana območja, saj se niso povezana območja, ki so črke prekrivala na Sliki 2.9, praktično nič spreminjala skozi zaporedne korake upragovanja.

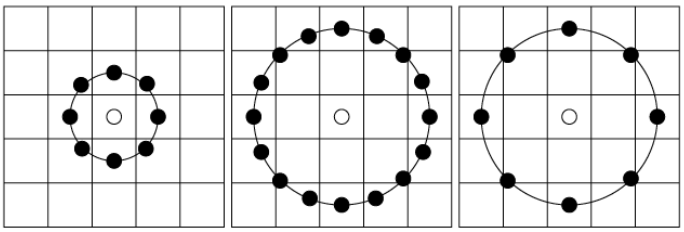
(a) Vhodna slika S.

(b) Rezultat detekcije maksimalno stabilnih regij.

Slika 2.10: Algoritem pravilno prepozna vse tri črke na sliki, saj se niso povezana območja, ki so črke prekrivala na Sliki 2.9, praktično nič spreminjala skozi zaporedne korake upragovanja.

2.6 Lokalni binarni vzorci

Lokalni binarni vzorci (angl. local binary pattern, LBP) je princip opisovanja lastnosti posamezne celice v sliki S, na podlagi razmerja do njenih sosedov oz. do celic, ki jih relacija sosednosti definira kot sosednje. Temelji na predpostavki, da ima tekstura, gledana lokalno, dve glavni lastnosti: vzorec ter intenziteto [6, 7].



Slika 2.11: Primeri definicije relacije sosednjosti. Vrednost točke, ki se ne nahaja v sredini celice se izračuna z interpolacijo. Slika vzeta iz [2].

Podobno kot MSER je mogoče tudi LBP lepo opisati po korakih:

1. Najprej določimo relacijo sosednosti, tako za vsako celico dobimo 8 sosedov (to število je seveda odvisno od relacije sosednosti, a v naslednjih korakih bom za olajšanje razlage uporabljal osnovno relacijo sosednosti, z osmimi sosedi) Primeri relacije sosednosti so prikazani na Sliki 2.11, z belo točko je označena osrednja celica za katero računamo vrednost deskriptorja, s črnimi pa so označene točke, s katerimi osrednjo točko primerjamo. Vrednost črne točke, ki se ne nahaja v sredini celice se izračuna z interpolacijo.

2. Sosednje celice upragujemo s centralno celico.

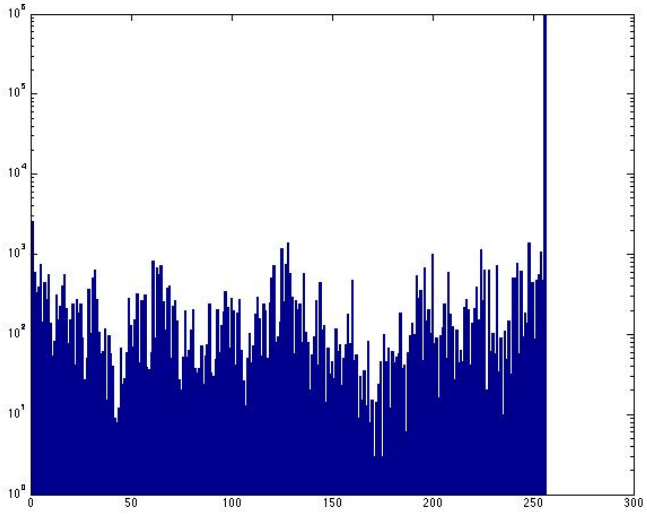
$$S = \begin{bmatrix} 7 & 13 & 2 \\ 1 & 5 & 9 \\ 4 & 2 & 8 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 0 \\ 0 & \bigcirc & 1 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow [11011000] \tag{2.9}$$

3. Na ta način ustvarimo 8-bitni vektor; dobljeni vektor sedaj rotiramo, dokler ne dobimo najmanjšega možnega binarnega števila. Na ta način ustvarimo deskriptor ki je odporen na rotacije slike.

$$[11011000] \Rightarrow [01101100] \Rightarrow [00110110] \Rightarrow [00011011] \equiv 5 \qquad (2.10)$$

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & \bigcirc & 1 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow [00011011] \Leftarrow \begin{bmatrix} 0 & 0 & 1 \\ 0 & \bigcirc & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad (2.11)$$

4. Lokalni deskriptorji združeni v vektor pa predstavljajo deskriptor celotne slike. Primer deskriptorja za naše višinske mate je prikazan na Sliki 2.12. Izrazita gruča na desni predstavlja površine, ki nas ne zanimajo, ostale gruče pa predstavljajo interesna območja:jarke,vrtače,itd. Na sliki je os Y prikazana v logaritemski skali, saj je gruča na desni tako izrazita, da ostalih vrednosti pri normalni skali skorajda ni moč opaziti.



Slika 2.12: Standardna oblika deskriptorja višinske mape. Za lažjo predstavo je uporabljena logaritemska skala na Y osi.

Osnovna oblika algoritma je bila predlagana in opisana leta 1996 [23]. Takrat je bila uporabljena osnovna oblika relacije sosednosti, za sosede je bilo določenih osem najbližjih celic, kjer je imela vsaka sosednja celica definirano fiksno utež:

$$\begin{bmatrix} 1 & 2 & 4 \\ 128 & \bigcirc & 8 \\ 64 & 32 & 16 \end{bmatrix}. \tag{2.12}$$

Nato pa je bil leta 2002 [22] nadgrajen, s čimer je bila dodana podpora za dodatne relacije sosednjosti ter odpornost na rotacije slike.

Danes igra LBP pomembno vlogo pri programih za detekcijo in določanje značilnosti obrazov [11], s predpostavko da je obraz neka vrsta teksture, lahko za klasifikacijo uporabljamo razširjeni algoritem LBP, ki uporabi deskriptorje pridobljene z večimi različnimi relacijami sosednosti, namesto veliko bolj kompleksnih algoritmov. Za doseganje boljših rezultatov se slika pri detekciji obrazov navadno razdeli na manjše regije, nato pa se lokalni deskriptorji vseh regij zložijo v skupni vektor, ki predstavlja deskriptor LBP za obraz.

2.7 Sobelov filter za detekcijo robov

Sobelov algoritem spada v skupino algoritmov za detekcijo robov, ki izvajajo detektcijo robov na podlagi aproksimacije gradientov v sliki. Tip filtra sta leta 1968 predstavila Irwin Sobel in Gary Feldman [31]. Podobni flitri, ki aproksimiraj gradiente so še Prewittov (P), Robertsov(R) in Scharrov(Sc).

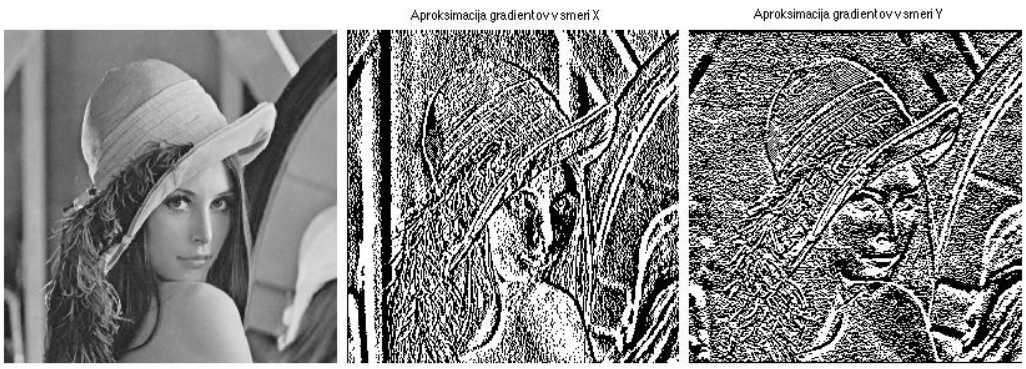
$$P = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \; R = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \; Sc = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \tag{2.13}$$

Sobelov algoritem uporabi dva filtra za konvolucijo,opisana v Poglavju 2.3, ki sta izbrana tako, da aproksimirata gradiente v sliki S, v navpični ter

vodoravni smeri

$$G_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \; G_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.14}$$

Dobljeni sliki sta tako odvoda slike v X in Y smeri (2.13). Nato se izračuna



Slika 2.13: Predstavitev gradientov v sliki v smereh X inY. V svetlih točkah je vrednost gradienta visoka, gre za hitro spremembo intenzitete.

skupni gradient slike, kjer se smerna odvoda seštejeta po naslednji enačbi:

$$G = \sqrt{G_x^2 + G_y^2}, \tag{2.15}$$

$$G = |G_x| + |G_y|. \tag{2.16}$$

(2.15) je osnovna enačba predstavljena v teoriji, medtem ko je (2.16) uporabljena kot njena aproksimacija, ki se uporablja v praksi. Svetla območja, ki

na sliki označujejo visoke vrednosti gradientov, lahko nato z upragovanjem ločimo od ostalih elementov na sliki. Primer je prikazan na Sliki 2.14, kjer je na levi predstavljena slika z skupnim gradientom, na desni pa rezultat upragovanja, kjer so bila pozitivna območja po upragovanju stanjšana na debelino ene celice. Povezana območja se na debelino ene celice stanjšajo tako, da algoritem iz njihovega roba odstranjuje celice toliko časa, dokler s tem ne bi prekinili prej povezanega območja.

(a) Gradienti sešteti po enačbi (2.15).

(b) Robovi v sliki označeni kot pozitivni elementi v binarni sliki po upragovanju in tanjšanju na debelino ene celice.

Slika 2.14: Predstavitev seštevanja gradientov ter označevanja robov na podlagi upragovanja.

Poglavje 3

Naš pristop k detekciji jarkov

Tekom izdelave diplomske naloge smo razvili naš algoritem za detekcijo pehotnih jarkov na podlagi višinskih map terena. Algoritem kot vhodni podatek dobi pripravljeno višinsko mapo, ter na podlagi višinske mape vrne binarno masko, kjer pozitivna območja predstavljajo jarke, hkrati pa binarna maska vsebuje tudi koordinate posameznih območji, ki jih umeščajo v državni koordinatni sistem Slovenije.
V tem poglavju bomo predstavili naš pristop detekcije jarkov. Celoten postopek je bil izveden po korakih in tako so razdeljena tudi podpoglavja. V Podpoglavju 3.1 bomo predstavili, kako smo se na podlagi principov opisanih v Podpoglavju 2 lotili glajenja terena ter pripravi višinskih standardiziranih višinskih map. Podpoglavje 3.2 opisuje pripravo terenskih značilnic z uporabo algoritmov opisanih v Podpoglavjih 2.4,2.5,2.6. V Podpoglavju 3.3 pa bomo predstavili naš pristop združevanja pridobljenih značilnic v končni rezultat našega algoritma.

3.1 Glajenje terena

Prvi korak pri izdelavi našega algoritma je bila priprava višinskih map, ki jih bomo uporabljali v nadaljnih korakih našega algoritma. Začeli smo z višinskimi mapami, ki smo jih pripravili v programu Manifold, s postopkom

opisanim v Poglavju 2.2. Težava je v tem, da je taka višinska mapa lahko močno razgibana in kot taka neprimerna za takojšnjo uporabo. Terena, ki sta si vizualno lahko zelo podobna, se bosta razlikovala po nadmorskih višinah, velik vpliv pa imajo tudi makro-topološke značilnosti:hribi, doline, rečna korita, itd.
Razvili smo algoritem za standardizacijo višinskih map, z uporabo konvolucije, opisane v Poglavju 2.3, kjer smo se znebili makro-topoloških značilnosti terena ter nadmorske višine, tako da je bila najnižja točka naše višinske mape 0. Na višinski mapi S smo tako uporabili naslednji pristop:

1. S smo duplicirali v $S1$ in $S2$, $S1$ nam bo služil kot osnova, nad $S2$ pa bomo izvajali konvolucijo.

2. Izbrali smo konvolucijski filter t med filtri predstavljenimi v Poglavju 2.3, ter po enačbi (3.1) izvedli konvolucijo nad $S2$. Na Sliki 3.1 je prikazan rezultat glajenja višinske mape s cikličnim uniformnim filtrom velikosti 5×5. Opazimo lahko, da so bile vse mikro-topološke značilnosti terena zglajene.

$$c(n_1, n_1) = \sum_{k_1=0}^{r-1} \sum_{k_2=0}^{c-1} S2(k_1, k_2) t(n_1 - k_1, n_2 - k_2) \tag{3.1}$$

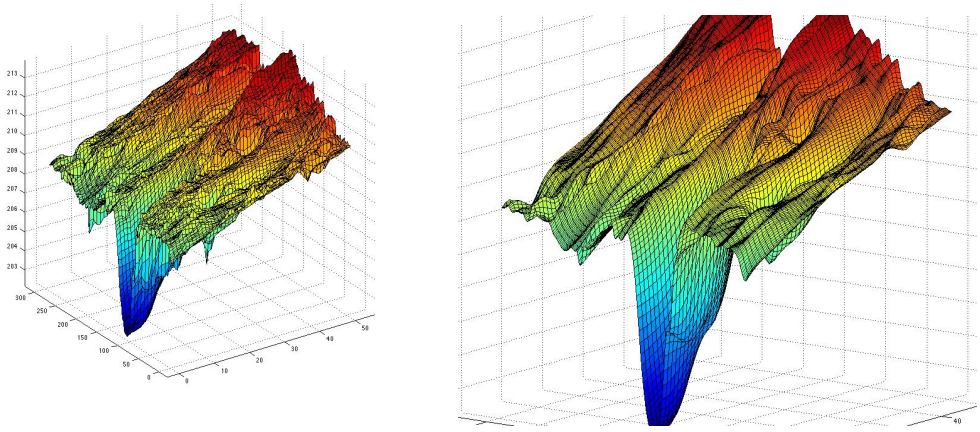
3. $S2$ smo nato odšteli od $S1$ ter tako dobili novo višinsko mapo Δ.

4. V zadnjem koraku smo višinsko mapo Δ še standardizirali na razpon vrednosti od 0 do 1, t.j.,

$$\Delta = \Delta + |Min(\Delta)|, \tag{3.2}$$

$$\Delta = \Delta / Max(Delta). \tag{3.3}$$

Pri nadaljnem razvoju algoritma smo upoabili rezultate pridobljene s konvolucijo z vsakim od konvolucijskih filtrov - Uniformni 3×3 filter, Gaussov 5×5 filter s standardno deviacijo $\sigma = 0.5$ ter CIkilični uniformni 5×5 filter,



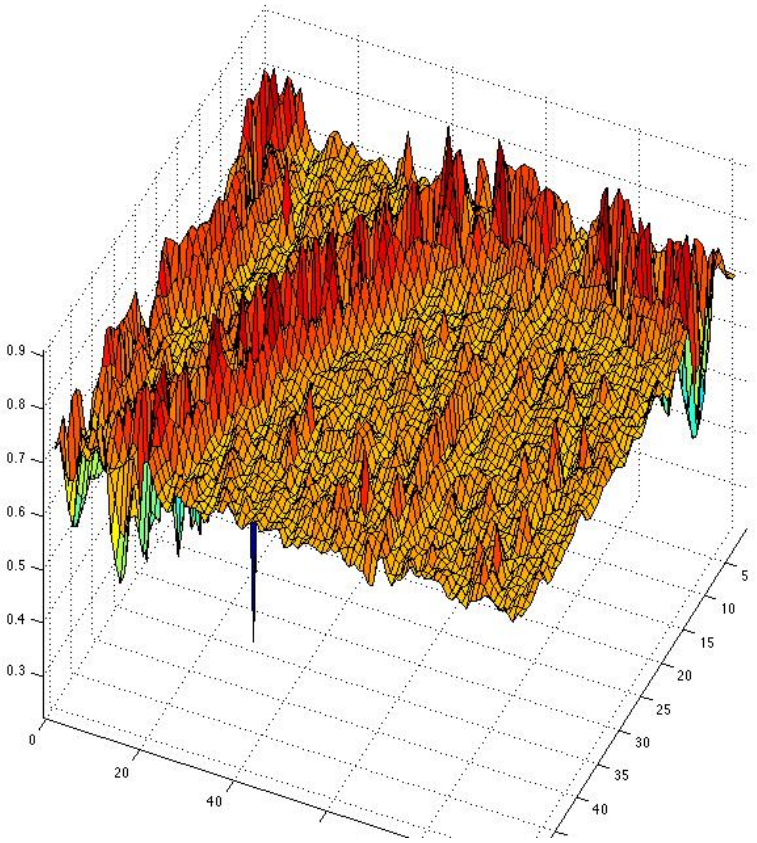
(a) Interpolirana višinska mapa. (b) Rezultat glajenja.

Slika 3.1: Prikaz glajenja s cikličnim uniformnim filtrom velikosti 5×5, vidimo lahko da so mikro-topološke značilnosti terena zglajene, prav tako tudi šum.

ki so predstavljeni v Poglavju 2.3. Glavno vlogo pa je v naslednjih korakih igrala višinska mapa Δ pridobljena s cikličnim uniformnim filtrom.
Na tako pridobljeni višinskih mapah, pa smo lahko v nadaljevanju uporabili algoritme in pristope za detekcijo regij opisane v Poglavju 2.

3.2 Priprava terenskih značilnic

Koraki v našem algoritmu do te točke so bili namenjeni pripravi podatkov za čim lažjo ter čim bolj učinkovito obdelavo, sami jarki pa so zaenkrat vidni še vedno samo s prostim očesom. Cilj priprave terenskih značilnic je torej obdelava podatkov, tako da jarkom oz. iskanim mikro-topološkim značilnostim priredim določene vrednost, na podlagi katerih jih lahko nato ločim od vsega ostalega. Na tej točki smo preiskusil veliko število pristopov in algoritmov, ter nato izbral tri, opisane v Poglavju 2, ki so zagotavljali najboljše rezultate, in na podlagi katerih smo nato nato nadaljevali do končnih rezultatov:



Slika 3.2: Standardizirana višinska mapa za območje na sliki 3.1. Pehotni jarki so vidni kot povezana ugreznjena območja, z izrazitimi robovi, na sliki obarvani temno rdeče.

Dinamično nastavljanje pragovne vrednosti

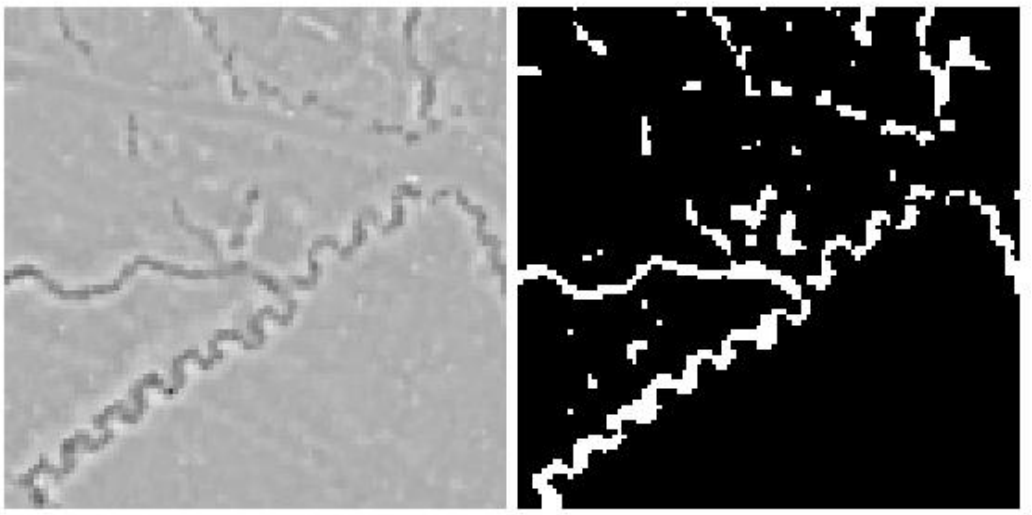
Prvo od treh značilnic smo pridobili z dinamičnim nastavljanjem pragovne vrednosti, saj ima višinska mapa po glajenju terena nekaj lepih lastnosti. Zaradi pravilne izbire filtra so sedaj na višinski mapi zelo izrazito vidne vse mikro-topološke značilnosti terena, z določeno količino šuma, celice, kjer pa mikro-topoloških značilnosti ni bilo pa imajo vse zelo podobne vrednosti (Slika 3.2). Taka razporeditev vrednosti, se zelo lepo pokaže tudi na hi-

stogramu višinske mape, kjer opazimo zelo izrazito gručo točk, ki predstavlja območje, ki nas ne zanima, ter nekaj manjših vrhov, ki predstavljajo različne mikro-topološke značilnosti ter šum. Z uporabo trikotniške metode za določanje pragovne vrednosti smo tako za vsako višinsko mapo lahko določil pragovno vrednost, ki je bila postavljena na vznožje največje gruče v histogramu (Poglavje 2.4). Na tako pridobljeni značilnici, kjer so celice katerih vrednost je manjše od pragovne nastavljene na 1, ter ostale na 0, so jarki in vrtače zelo lepo vidni, a hkrati opazimo tudi veliko količino šuma. Primer si lahko ogledamo na Sliki 3.3. Na levi je prikazanja standardizirana višinska mapa, na desni pa binarna slika pridobljena z dinamičnim nastavljanjem pragovne vrednosti. Jarki so pravilno označeni, ceste algoritem ne označi, opazimo pa lahko šum. Za lažjo vizualizacijo je prikazan del višinske mape velikosti $100m \times 100m$, medtem ko algoritem operira z višinskimi mapami velikosti $1000m \times 1000m$.

Detekcija maksimalno stabilnih regij

Naslednji pristop je bila uporaba algoritma za detektiranje maksimalno stabilnih regij. Izkazalo se je da je glavna prednost algoritma MSER ta, da se izogne šumu, ki je prisoten v značilnici, pridobljeni z dinamičnim nastavljanjem pragovne vrednosti (Slika 3.3). Hkrati pa s spreminjanjem parametra maksimalne dopuščene variacije, gre za dopuščeno variacijo med povezanim območjem ter njegovo okolico, lahko bolj natančno prepoznamo tudi vrtače, kar je tudi eden od stranskih ciljev naše diplomske naloge, s katero se bomo ukvarjali tudi v prihodnje. Primer rezultata algoritma MSER je prikazan na Sliki 3.4. Ker gre za problematično območje lahko hkrati vidimo prednosti in slabosti algoritma MSER v primerjavi z metodo z dinamičnim nastavljanjem pragovne vrednosti. Vidimo, da je na sliki manj šuma, vendar so pozitivna območja mnogo širša od jarkov, nekateri od jarkov pa so izpuščeni. Na zgornjem območju so jarki namreč šibko definirani, zato pride do določene količine razlitja območji.

Jarki so od mikro-topoloških značilnosti najbolj izraziti, saj imajo bolj ali

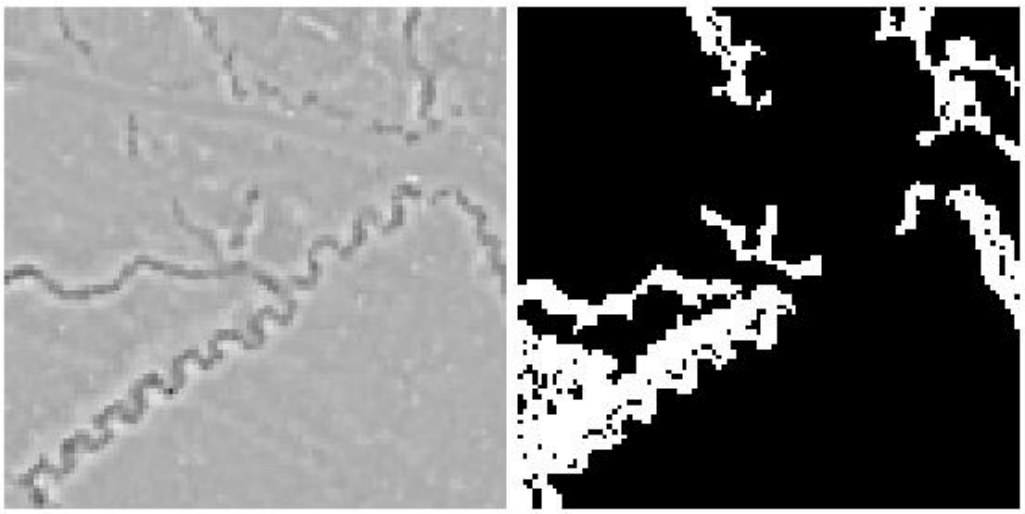


Slika 3.3: Standardizirana višinska mapa na levi, na desni pa je binarna slika pridobljena z dinamičnim nastavljanjem pragovne vrednosti. Jarki so pravilno označeni, ceste algoritem ne označi, opazimo pa lahko šum. Za lažjo vizualizacijo je prikazan del višinske mape velikosti $100m \times 100m$, medtem ko algoritem operira z višinskimi mapami velikosti $1000m \times 1000m$.

manj navpično odrezane stranice, tako da jih algoritem MSER zazna že pri zelo nizki stopnji dopuščene variacije območja, ko pa to vrednost postopoma povečujemo, bodo na sliki najprej prepoznane še vrtače, šibkeje definirani jarki Z nadaljnim dvigovanjem dovoljene variacije, pa se bodo detektirana območja razširila čez celotno sliko in rezultati bodo postali neuporabni.

Uporaba lokalnih binanih vzorcev

Tretji pristop, ki smo ga ubrali, pa je bila uporaba algoritma za detekcijo lokalnih binarnih vzorcev. Na enaki višinski mapi, iz katere smo pridobil

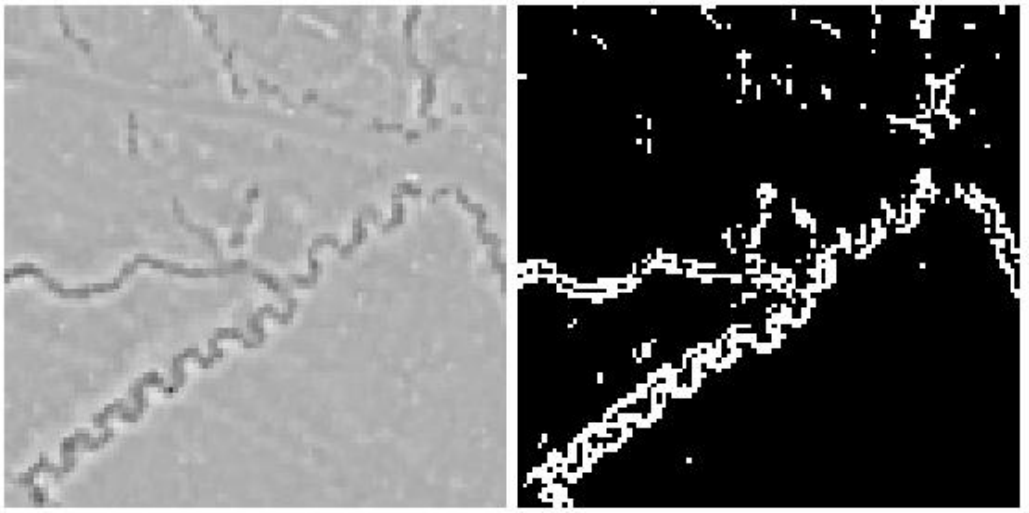


Slika 3.4: Prikaz delovanja algoritma MSER na enakem območju kot na Sliki 3.3. Ker gre za problematično območje lahko hkrati vidimo prednosti in slabosti algoritma MSER v primerjavi z metodo z dinamičnim nastavljanjem pragovne vrednosti. Vidimo, da je na sliki manj šuma, vendar so pozitivna območja mnogo širša od jarkov, nekateri od jarkov pa so izpuščeni. Na zgornjem območju so jarki namreč šibko definirani, zato pride do določene količine razlitja območji.

prejšnji dve značilnici algoritem, LBP ni vračal zadovoljivih rezultatov, zato smo tu potreboval vmesni korak, v katerem smo iz višinske mape ustvaril mapo gradientov. To smo storil s pomočjo Sobelovih operatorjev, s katerima smo dobil smerna gradienta slike v x - G_x in y - G_y smeri, ter ju nato združil v skupni gradient po naslednji formuli:

$$G = \sqrt{(G_x^2 + G_y^2)}. \tag{3.4}$$

Postopek je bolj podrobno opisan v Poglavju 2.7. Na tako pridobljeni mapi gradientov pa je nato algoritem LBP vrnil rezultate, ki so bili primerljivi s tistimi pridobljenimi po metodi dinamičnega nastavljanja pragovne vrednosti. Po uporabi algoritma LBP iz višinske slike dobimo matriko, katere



Slika 3.5: Z upragovanjem deskriptorja LBP dobimo zgoraj prikazano binarno sliko. V naslednjih korakih razvoja algoritma slik pridobljenih z upragovanjem LBP deskriptorja nismo uporabil, saj niso prinesle nobenih izboljšav, a bodo uporabljene v nadgradnji našega algoritma v prihodnosti.

celice zavzamejo vrednosti od 0-255; bolj podrobni opis se nahaja v Poglavju 2.6, nato pa smo nad dobljeno matriko ponovno uporabili trikotniško metodo za določanje pragovne vrednosti, Poglavje 2.4, saj je histogram tako dobljene matrike podoben tistemu iz Poglavja 3.2, tako da metoda ponovno vrne primerno vrednost za upragovanje. Primer tako izračunane značilnice je predstavljen na Sliki 3.5.

3.3 Končna obdelava značilnic

Z vsemi tremi pripravljenimi značilnicami je sledila njihova obdelava ter združevanje v končno binarno sliko. Tu si z značilnico pridobljeno z algoritmom LBP nismo mogli pomagati, saj so pozitivna območja na njej preveč raztrgana (Slika 3.5), tako da smo se posvetili obdelavi značilnic pridobljenih z algoritmom MSER ter z dinamičnim nastavljanjem pragovne vrednosti. Začeli smo z morfološkimi operacijami, s katerimi smo se želeli znebiti šuma ter poudarit jarke oz. tista območja, ki jih imamo v interesu.

3.3.1 Odstranjevanje regij glede na površino

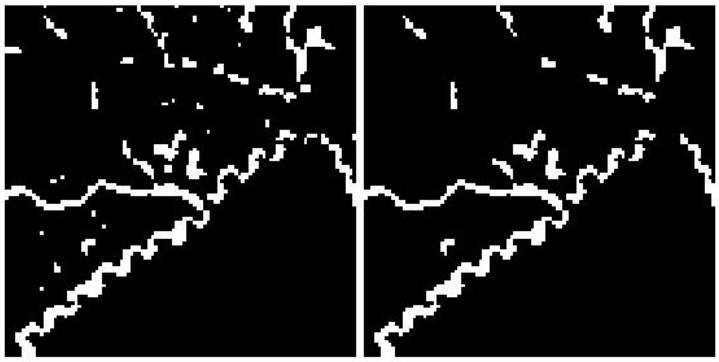
Definirali smo parameter najmanjše sprejemljive velikosti: γ, za povezano območje, nato pa smo iz značilnice S odstranili vsa povezana območja: s katerih površina je bila manjša od γ, tako da smo njihove celice nastavili na 0, t.j.,

$$\forall s \in S : \begin{cases} s = s & ; A(s) > \gamma \\ s = 0 & ; A(s) \leq \gamma \end{cases}. \tag{3.5}$$

Pri čemer $A(s)$ predstavlja funkcijo, ki prešteje kolikšno število celic območje vsebuje.

3.3.2 Razširjanje in povezovanje regij

Na Sliki 3.6 opazimo, da so območja, ki označujejo jarke dostikrat pretrgana, zato smo s spodnjimi koraki poskusili taka območja pred nadaljno obdelavo povezati. Postopek je prikazan na Sliki 3.7, kjer so predstavljeni trije zaporedni koraki razširjanja in povezovanja. Začnemo z osnovno sliko, algoritem povezana območja v njej razširi, nato poskuša povezati območja, ki so si dovolj blizu, v zadnjem koraku pa povezana območja stanjša na prvotno debelino. Najprej smo območje razširili, tako da se vsakemu povezanemu območju doda po eno celico na rob območja, a samo pod pogojem da s tem ne



Slika 3.6: Primer odstranjevanja regij za $\gamma = 10$.

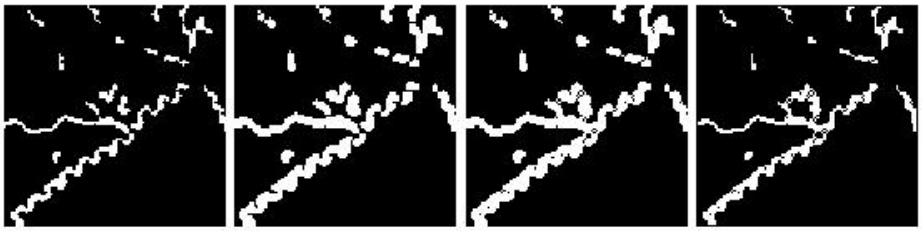
povežemo prej nepovezanih regij. Nato smo uporabili operacijo premoščanja, ki deluje tako, da celico z vrednostjo 0 nastavi na 1, če ima dva pozitivna soseda, ki nista soseda med seboj. S tem smo uspešno povezali območja, ki so bila dovolj blizu skupaj (3m oz. 3 celice), da smo lahko predvidevali, da pokrivajo isti jarek.

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.6}$$

Nazadnje pa smo območje stanjšali, tako da smo uporabili obratni algoritem od tistega uporabljenega v Poglavju 3.3.2, ki je povezanim območjem odstranil po eno celico z roba območja, tako kot je to prikazano na Sliki 3.7.

3.3.3 Ločevanje na podlagi ekscentričnosti

Po morfoloških operacijah, so na obeh značilnicah (MSER, dinamično upragovanje) ostala večinoma le območja, ki predstavljajo jarke ter vrtače oz. to je bila naša predpostavka, tako da smo potrebovali lastnost na podlagi katere



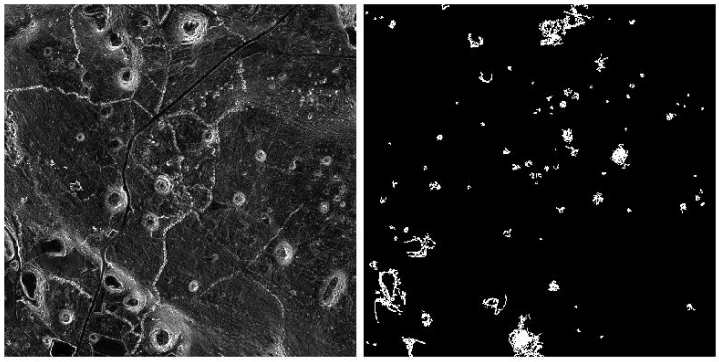
Slika 3.7: Primer povezovanja bližnjih regij. Prikazani so trije zaporedni koraki razširjanja in povezovanja. Začnemo z osnovno sliko, algoritem povezana območja v njej razširi, nato poskuša povezati območja, ki so si dovolj blizu, v zadnjem koraku pa povezana območja stanjša nazaj na prvotno velikost.

bi jih ločili med seboj.
Ekscentričnost je definirana z naslednjo enačbo:

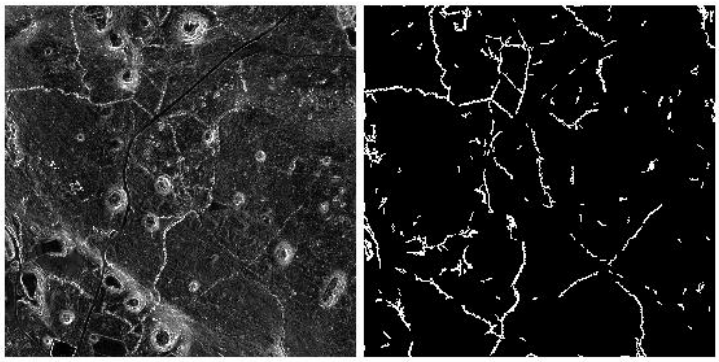
$$\sqrt{1 - \frac{b^2}{a^2}}, \tag{3.7}$$

kjer a in b predstavljata polovici dolžine daljše ter krajše osi elipse, ekscentričnost pa zavzame vrednosti na intervalu: $e \in [0, 1]$, kjer ima krožnica ekscentričnost 0, pri paraboli pa zavzame vrednost 1.
S pomočjo ekscentričnosti nam je nato uspelo ločiti vrtače od jarkov. Tu smo izkoristili prednost algoritma MSER, ki je vrtače označil dosti bolje, tako da smo le-te s pomočjo ekscentričnosti poiskali v značilnici pridobljeni z algoritmom MSER. Na Sliki 3.8 je prikazan rezultat detekcije vrtač na podlagi ekscentričnosti, na Sliki 3.9 pa rezultat detekcije jarkov, prav tako na podlagi

ekscentrčnosti.



Slika 3.8: Detektirane vrtače na podlagi značilnice, pridobljene z algoritmom MSER; označene so samo tiste, ki se z jarki ne prekrivajo.



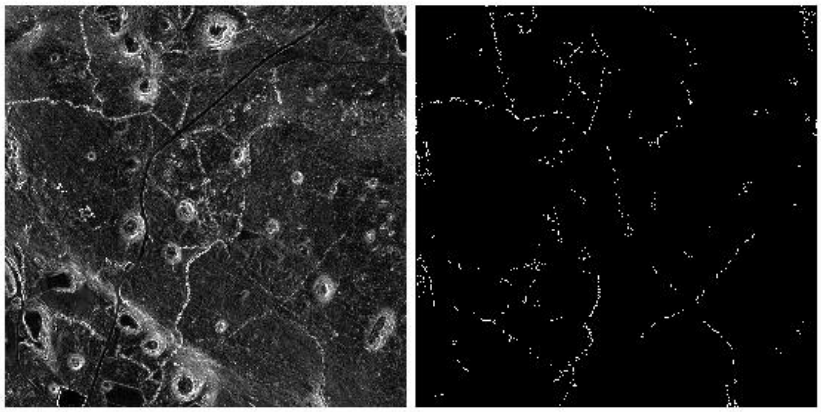
Slika 3.9: Detektirani jarki na podlagi značilnice, pridobljene z algoritmom za dinamično nastavljanje pragovne vrednosti. Še vedno opazimo visoko število nepravilno prepoznanih območji; ta bomo odstranili s pomočjo detektiranih vrtač na Sliki 3.8.

3.3.4 Zaključna obdelava

Na podlagi Slik 3.8, 3.9, smo sedaj ustvariti končni rezultat našega algoritma. Najprej smo uporabili algoritem iz Poglavja 3.3.2 za tanjšanje povezanih območji ter z njim stanjšali detektirane vrtače s Slike 3.8. Z $S1$ označimo značilnico z jarki ter z $S2$ značilnico z vrtačami, z n pa število vrstic oz. stolpcev v slikah $S1$ in $S2$, tako smo skupno značilnico T dobili na sledeči način:

$$\forall i,j \in [1,n] : T(i,j) = \begin{cases} S1(i,j) & ; S2(i,j) == 0 \\ S1(i,j) \oplus S2(i,j) & ; S2(i,j) == 1 \end{cases}. \tag{3.8}$$

Tako smo iz značilnice $S1$ odstranili vsa napačno prepoznana območja, ki so jih povzročile vrtače. Nad T smo nato ponovno pognali morfološke algoritme iz Poglavja 3.3.1 in 3.3.2, da smo se znebili novonastalega šuma ter povezali območja, katera smo v vmesnih korakih morda prekinili. Na koncu pa smo nad T pognali algoritem za tanjšanje povezanih območji (Poglavje 3.3.2), s katerim smo želeli stanjšati povezana območja na debelino ene celice. Algoritem tako odstranjuje celice z roba povezanega območja, dokler tega ni več moč storiti, ne da da bi povezano območje prekinili. Končni rezultat postopka je prikazan na Sliki 3.10, kjer smo na podlagi zgoraj opisanih postopkov združili binarni sliki predstavljeni na Sliki 3.8 in Sliki 3.9.



Slika 3.10: Na levi je prikazana višinska mapa, na desni pa izhodni rezultat našega algoritma za območje velikosti $1km \times 1km$.

Poglavje 4

Eksperimentalna analiza

Poglavje je razdeljeno na dva dela, v prvem bomo predstavili parametre našega algoritma,zbirko podatkov na kateri smo naš algoritem testirali ter cilje, katerih ciljne vrednosti smo z našim algoritmom želeli doseči. V drugem delu pa bomo predstavili in ovrednotil naše rezultate ter jih primerjali z referenčno metodo.

4.1 Parametri

V tem poglavju si bomo ogledali, kateri so glavni parametri, ki so nastopali v našem algoritmu, na podlagi česa smo izbrali njihove vrednosti ter predstavili njihov vpliv na naš algoritem.

Izdelava višinske mape iz točkovnega oblaka

Sistem Lidar je podatke o terenu vrnil v obliki točkovnega oblaka, iz katerega smo želeli izdelati višinske mape terena nad katerimi bi nato pognali naš algoritem. Pri tem je bilo potrebno izbrati velikost celice naše višinske mape, ki bo omogočala kar najboljše rezultate. Gostota točk v točkovnem oblaku je v povprečju znašala 5 točk na m^2, višina celice v višinski mapi pa je določena z interpolacijo, za kar je poskrbel sistem Manifold. Testirali smo celice z dolžinami robov: 0.5m, 1m, 2m, 3m, 5m in 10m. Pri velikosti celice

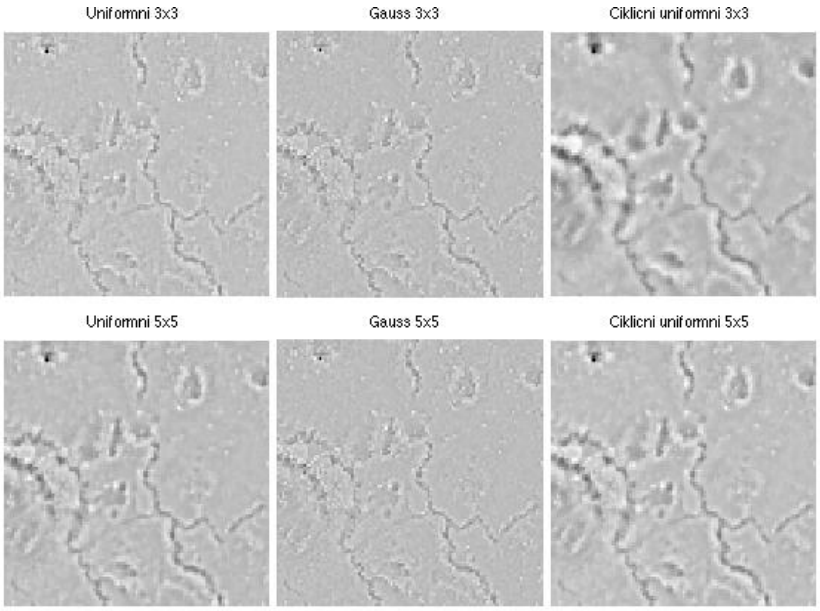
$0.5m \times 0.5m$ je šlo za premajhne celice, kar je posledično na višinski mapi povzročilo preveč šuma, s celicami z robom 2m in več, pa je bila že na pogled očitna izguba mikro-topoloških značilnosti terena. Celica velikosti $1m \times 1m$ je bila izbrana kot najbolj primerna. Vpliv velikosti celice na lastnosti terena je viden na Sliki 2.4, kjer so prikazani rezultati za celice velikosti 1m, 2m in 3m. Pri celicah z velikostjo nad 3m je nato prišlo le še do večjih izgub mikro-topološkiih značilnosti.

Glajenje terena s konvolucije

Pri glajenju terena, opisanem v Poglavju 2.3, smo se odločali med tipi filtrov ter njihovimi dimenzijami. Najprej smo vizualno pregledali rezultate izračunane s filtri, ki jih podpira program Matlab ter na podlagi tega izbrali tri filtre, ki jih bomo še dodatno testirali, tako da bomo lahko določili kateri od njih je najbolj primeren ter tudi katera dimenzija filtra vrne najbolj primerne rezultate. Filtri, ki smo jih dodatno testirali so: uniformni, Gaussov ter ciklični uniformni. Rezultati, pridobljeni z vsakim od njih, pa so predstavljeni na Sliki 4.1. Vsakega od filtrov smo nato testirali z različnimi dimenzijami ter za vsako od njih izračunali terenske značilnice. Za potrebe našega algoritma se je najbolje odrezal Cikilčni uniformni filter velikosti 5×5

Minimalna povezana območja

Zaradi velike količine šuma, ki je bil prisoten skozi vse korake našega algoritma, smo vpeljali parameter, ki določa minimalno površino povezanega območja γ. Parameter smo testirali na naši testni množici in opazovali, kakšna je maksimalna vrednost γ, pri kateri se znebimo samo šuma in ne tudi pravilno prepoznanih regij. Izkaže se, da so v veliki večini primerov, razen v kritičnih primerih, opisanih v Poglavju 4.5, pravilno prepoznana območja večja od 60 celic, γ pa je v našem algoritmu nastavljen na $\gamma = 40$, tako da ne bi po nepotrebnem odstranjevali pravilno prepoznanih območji.



Slika 4.1: Rezultati konvolucije s tremi različnimi filtri: uniformni, Gaussov ter ciklični uniformni. Za vsak filter sta predstavljena rezultata za velikosti filtra 3×3 ter 5×5.

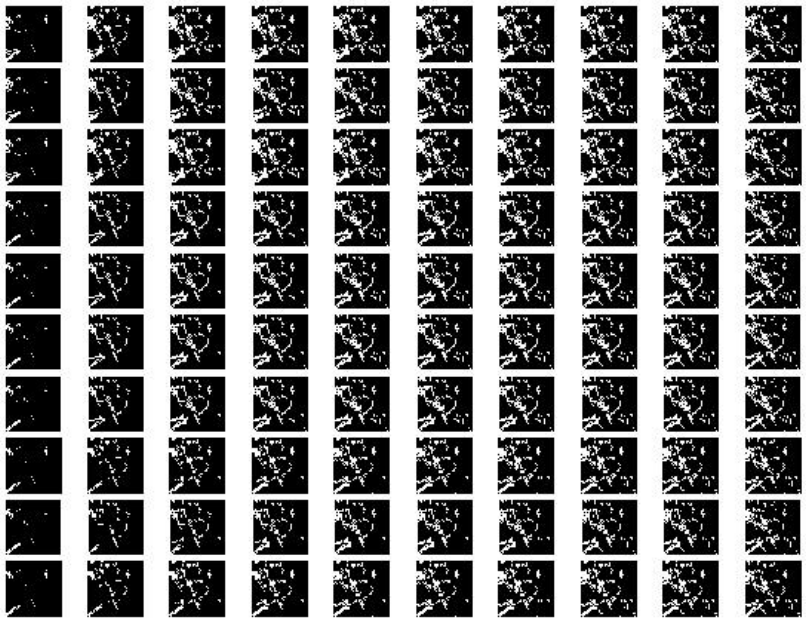
Detekcija maksimalno stabilih regij

Pri uporabi algoritma za detekcijo maksimalno stabilnih regij (MSER), opisanega v Poglavju 2.5, smo nastavljali naslednja dva parametra: korak upragovanja: δ ter maksimalno dopuščeno variacijo povezane regije: σ. Parametra sta opisana v Poglavju 2.5, vpliv spreminjanja δ in σ na detektiranje regij pa je prikazan na Sliki 4.2. Na Y osi, je predstavljen parameter δ, ki zavzame vrednosti od 0.1 do 1, na X osi pa je predstavljen parameter σ, ki zavzame vrednostiod 0.05 do 0.5. i=delta j=sigma

Parameter δ, ki pove, kolikšen odstotek vrednosti razpona naših podatkov naj algoritem vzame kot korak pri upragovanju, smo nastavili na $\delta = 0.005$. Parameter σ, ki označuje maksimalno dopuščeno variacijo, pa smo nastavili na vrednost $\sigma = 0.1$.

Oba parametra sta nastavljena tako, da kar najbolj selektivno izločata neprimerne regije, saj smo za jarke lahko z veliko verjetnostjo predpostavili,

da so zelo izrazito definirani s praktično navpičnimi robovi. Tako smo se izogniili napačnim identifikacijam območji, vendar pa je algoritem na račun tega zavrnil določena območja, ki pa dejansko predstavljajo jarek. To se je dogajalo predvsem v kritičnih primerih, opisanih v Poglavju 4.5



Slika 4.2: Vpliv spreminjanja parametrov δ in σ. Na Y osi, je predstavljen parameter δ, ki zavzame vrednosti od 0.1 do 1, na X osi pa je predstavljen parameter σ, ki zavzame vrednostiod 0.05 do 0.5.

Ekscentričnost

Kot smo opisali v Poglavju 3.3.3, smo računanje ekscentričnosti v našem algoritmu uporabili v dveh primerih. Na terenski značilnici, pridobljeni z algoritmom MSER, smo na podlagi ekscentričnosti elips, ki so orisovala povezana območja, detektirali vrtače, na terenski značilnici pridobljeni z dinamičnim upragovanjem, pa jarke. Na podlagi testiranj, opravljenih na terenskih značilnicah naših testnih podatkov, predastavljenih v Poglavju 4.2, smo uspeli izračunati okvirne vrednosti praga ekscentričnosti - Δ za vrtače in jarke, ki so kar najbolj pravilno klasificirale povezana območja.

Tabela 4.1: Pragovne vrednosti ekscentričnosti

Značilnica	**Prepoznava območji**	**Interval pragovne vrednosti**
Pragovna vrednost	Prepoznava jarkov	$\Delta > x \in [0.85, 0.92]$
MSER	Prepoznava vrtač	$\Delta < x \in [0.7, 0.8]$

Tabela 4.1 prikazuje izračunane pragovne vrednosti za Δ, v primeru detekcije jarkov, lahko za najverjetnejša kandidate določimo tista povezana območja za katera velja:

$$\Delta \geq x \in [0.85, 0.92]. \tag{4.1}$$

V primeru detekcije vrtač, pa za najverjetnejše kandidate določimo tista povezana območja za katera velja

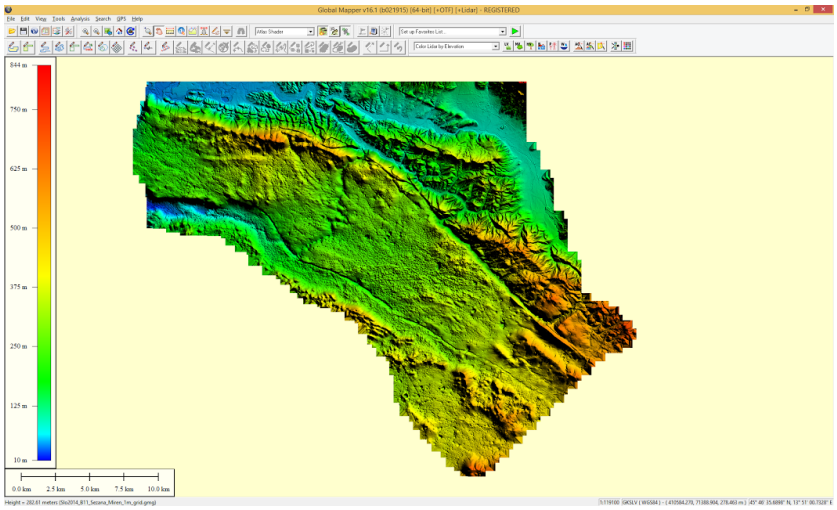
$$\Delta \leq x \in [0.7, 0.8]. \tag{4.2}$$

Maksimalna dopuščena oddaljenost

Pri ocenjevanju uspešnosti našega algoritma smo določili parameter maksimalne dopuščene oddaljenosti: β, s katerim smo uravnavali dopuščeno razdaljo med regijo, ki jo kot rezultat vrne naš algoritem in jarkom samim, na kateri se bo naša regija obravnavala kot pravilna. V našem primeru smo parameter β nastavili na $\beta = 10$; s tem smo kompenzirali odstopanja zaradi šuma ter izgube natančnosti, hkrati pa smo upoštevali dejstvo, da za naše potrebe ni smiselno zahtevati, da bodo jarki označeni do celice natančno. Tako nastavljen prag β pa je še vedno dovolj nizek, da kot napačna označi vse območja, ki so posledica ostalih mikro-topoloških značilnosti in ne jarkov.

4.2 Testni podatki in testiranje

Iz podatkov, ki so nam bili na voljo (Slika 4.3) ob začetku izdelave diplomske naloge, in ki so predstavljali Vipavsko dolino, smo za testiranje uspešnosti našega algoritma izbrali $4km^2$ terena.



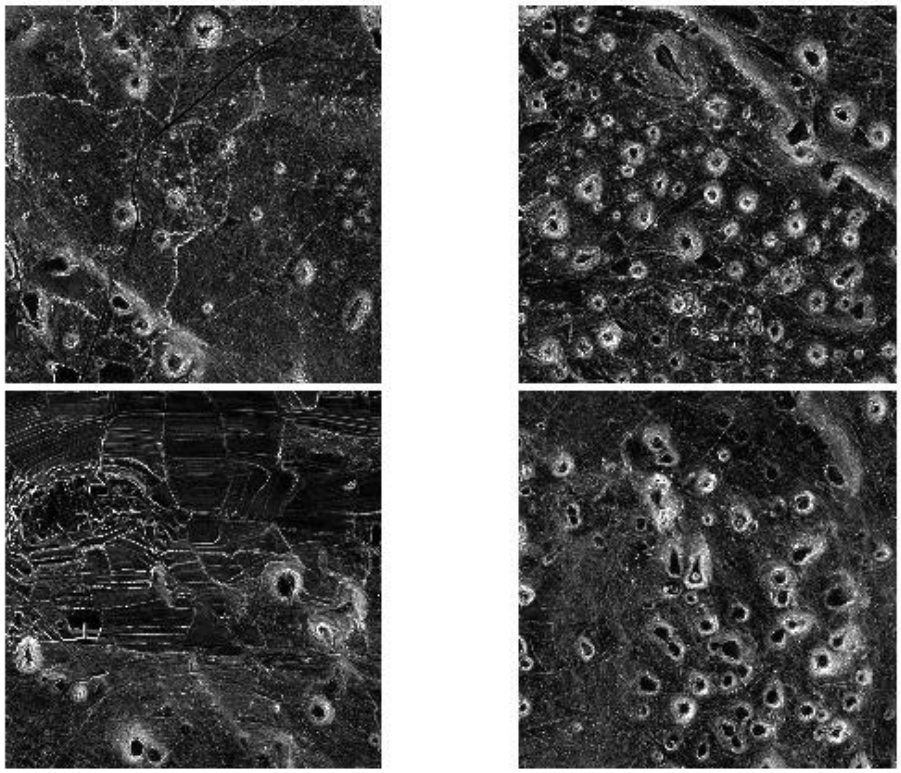
Slika 4.3: Predstavitev podatkov, ki smo jih imeli na voljo na samem začetku izdelave našega algoritma, v programu Manifold. Na sliki vidimo Vipavsko dolino.

Pri samem ocenjevanju uspešnosti delovanja našega algoritma smo se osredotočili na dva kriterija: uspešnost in zanesljivost. Želeli smo, da bi naš algoritem uspešno prepoznal kar največje število jarkov na sliki (uspešnost), hkrati pa bi vrnil kar najmanjši odstotek pozitivno označenih regij, ki pa ne predstavljajo jarkov (zanesljivost). Pri zastavljanju osnovnih mer uspešnosti, ki smo želeli z našim algoritmom doseči, smo veliko večji pomen namenili parametru zanesljivosti, saj je veliko bolj kot količina prepoznanih jarkov pomembno ali je pozitivno prepoznana regija res jarek. Algoritem z nizko zanesljivostjo nam namreč ne bi bil v veliko pomoč, četudi bi prepoznal sto odstotkov vseh jarkov,kjer bi le vsaka druga ali tretja pozitivna regija predstavljala jarek, ostale pa bi predstavljale šum. Osnovne mere uspešnosti smo si tako izbrali na sledeči način: uspešnost: Θ_1 ter zanesljivost: Θ_2:

$$\Theta_1 \geq 0.8, \tag{4.3}$$

$$\Theta_2 \geq 0.75. \tag{4.4}$$

Na zgoraj predstavljena testna območja (Slika 4.4) smo nato z roko vrisali vse



Slika 4.4: Predstavitev štirih višinskih map, velikosti $1km \times 1km$, ki so nam služile za ocenjevanje uspešnosti našega algoritma. Izbrane so bile tako, da so zajele kar najbolj raznolike terenske značilnosti, na katere lahko naletimo, več o tem v Poglavju 4.5.

jarke, ter s tem ustvarili binarne slike, s katerimi smo nato lahko primerjali naše rezultate.

4.2.1 Ocenjevanje zanesljivosti

Uporabili smo predpostavko, da bodo naši rezultati predstavljali vizualno pomoč pri proučevanju jarkov, zato od algoritma ni potrebno pričakovati, da bo jarke označil do celice natančno, temveč le, da je oznaka jarka dovolj blizu jarku samemu. Tako visoke natančnost prav tako ni bilo moč pričakovati, ker smo v vseh korakih priprave podatkov izgubili določeno količino natančnosti, ko smo podatke interpolirali. Na podlagi tega smo določili parameter dovo-

ljenega odstopanja rezultatov β, s katerim smo določili, kako daleč od jarka je lahko naša detektirana regija, da bo prepoznava še vedno obravnavana kot uspešna. Za naš rezultat Ω_1 in referenčno sliko Ω_2 tako velja:

$$\forall x \in \Omega_1 \wedge x == 1 : \begin{cases} x = 1 & ; \exists y \in N_\beta(x) \in \Omega_2 : y == 1 \\ x = 0 & ; sicer \end{cases}, \tag{4.5}$$

pri čemer $N_\beta(x)$ označuje oklolico celice x s polmerom β.
Preverjanje zanesljivosti na tak način pa bi bilo časovno zelo zahtevno, če bi naš algoritem spisali na podlagi Enačbe 4.5, zato smo ga implementirali na sledeči način.

1. Uporabili smo algoritem iz Poglavja 3.3.2 za razširitev povezanih območji in razširili Ω_2 za β v $\Omega_{2\beta}$. Na roko vrisana območja, ki so označevala jarke, so bila razširjena iz debeline ene celice na širino 2β celic.

2. Uporabili smo logični operator "in":

$$\Omega = \Omega_{2\beta} \wedge \Omega_1. \tag{4.6}$$

   Na ta način smo ohranili tista povezana območja, ki jih je kot rezultat vrnil naš algoritem in za katera velja, da je v njihovi okolici β vsaj ena celica, za katero smo na roko določili, da predstavlja jarek.

3. Prešteli smo število celic, ki jih algoritem primerjanja določi kot pozitivne prepoznave: Ω ter število celic, ki jih kot rezultat vrne naš algoritem za detekcijo jarkov: Ω_1.

$$\alpha_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} \Omega(i,j), \tag{4.7}$$

$$\alpha_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \Omega_1(i,j). \tag{4.8}$$

   Kjer n predstavlja število vrstic, oz stolpcev v obeh binarnih slikah. Ker gre za binarne slike, pozitivne celice enostavno preštejemo tako, da seštejemo vse elemente v sliki.

4. Zanesljivost smo izračunali kot razmerje med uspešnimi prepoznavami ter vsemi prepoznavami, ki jih vrne naš algoritem, t.j.,

$$\Theta_2 = \frac{\alpha_1}{\alpha_2}. \tag{4.9}$$

4.2.2 Ocenjevanje uspešnosti

Uspešnost našega algoritma smo testirali z enakim algoritmom, kot smo ga uporabili za računanje zanesljivosti (Poglavje 4.2.1), le da smo zamenjali vhodne podatke. Zanimalo nas je, kolikšen odstotek območji vrisanih na roko, je naš algoritem prepoznal. Z enako vrednostjo parametra β kot smo jo uporabili v Poglavju 4.2.1, smo to storili na sledeči način:

1. Uporabili smo algoritem iz Poglavja 3.3.2 za razširitev povezanih območji in razširili Ω_1 za β v $\Omega_{1\beta}$. Povezana območja, ki jih je kot rezultat vrnil naš algoritem za detekcijo jarkov in imajo debelino ene celice, so bila tako odebeljena na širino 2β celic.

2. Uporabili smo logični operator "in":

$$\Omega = \Omega_{1\beta} \wedge \Omega_2. \tag{4.10}$$

   Na ta način ohranimo tista na roko vrisana območja, ki imajo v okolici β vsaj eno celico, ki jo je kot pozitivno označil naš algoritem za detekcijo jarkov.

3. Prešteli smo število celic, ki jih algoritem primerjanja določi kot pozitivne prepoznave: Ω ter število celic, ki jih kot rezultat vrne naš algoritem za detekcijo jarkov: Ω_2.

$$\alpha_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}\Omega(i,j), \tag{4.11}$$

$$\alpha_2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\Omega_1(i,j). \tag{4.12}$$

4. Uspešnost smo izračunali kot razmerje med uspešno prepoznanimi območji vrisanimi na roko, prepoznavami ter vsemi območji, ki smo jih vrisali, t.j.,

$$\Theta_2 = \frac{\alpha_1}{\alpha_2}. \tag{4.13}$$

4.3 Predstavitev rezultatov

V tem poglavju bomo predstavili rezultate uspešnosti in zanesljivosti našega algoritma, izračunane po metodah opisanih v Poglavjih 4.2.1 in 4.2.2, hkrati pa bodo predstavljeni še rezultati, ki jih je na testni množici dosegel algoritem z uporabo Sobelovega filtra [31] za detekcijo robov. Metodo na podlagi Sobelovega filtra, predstavljen v Poglavju 2.7, smo izbrali za referenčno metodo, s katero bomo primerjali uspešnost našega algoritma. Metoda z uporabo Sobelovih filtrov je bila izbrana, saj primerljivih rezultatov, s katerimi bi lahko primerjali naše rezultate, za naš izbrano tematiko še ni bilo objavljenih. Uporaba Sobelovih filtrov je zelo razširjena, program Matlab jo ima implementirano, hkrati pa vrača dokaj solidne rezultate, kot bomo videli v nadaljevanju.

Za standardni tip terena je rezultat našega algoritma prikazan na Sliki 4.5. Na Sliki 4.6 pa so za lažjo vizualizacijo rezultati prikazani barvno. Z zeleno so tako označeni jarki, ki jih je naš algoritem pravilno označili, z rdečo so označeni jarki, ki jih naš algoritem ni označil, z rumeno pa so označena območja, katera je naš algoritem označil za jarke, a to niso.

Tabela 4.2: Uspešnost detekcije pehotnih jarkov

Teren	**Rezultat**	**Interval vrednosti**	**Sobel**
Srandardni	79,45 %	75-85 %	99,65 %
Problematični - vrtače	45,3 %	40-55 %	98,49 %
Problematični - človek	63,58 %	50-65 %	97,94 %
Problematični - erozija	18,48 %	10-25 %	97,72 %

Tabela 4.3: Zanesljivost detekcije pehotnih jarkov

Teren	**Rezultat**	**Interval vrednosti**	**Sobel**
Srandardni	72,72 %	65-85 %	42,04%
Problematični - vrtače	44,87 %	30-55 %	27,6 %
Problematični - človek	5,75 %	3-20 %	4,09 %
Problematični - erozija	21,56 %	10-25 %	15,5 %

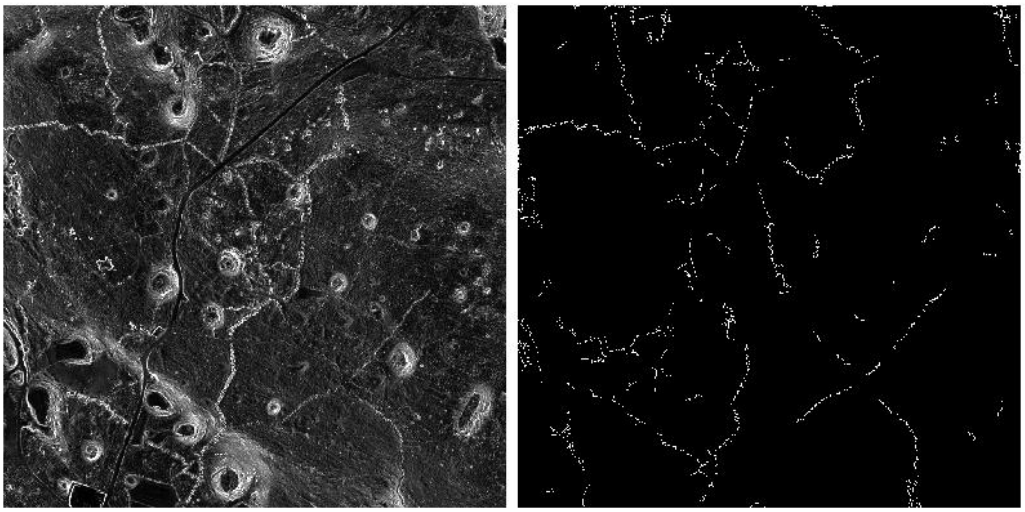
Kot smo opisali v Poglavju 4.2, smo za preverjanje uspešnosti in zanesljivosti našega algoritma izbrali $4km^2$ terena, tako, da smo zajeli kar najbolj raznolike terenske značilnosti, ki so imele velik vpliv na pridobljene rezultate in so opisane v Poglavju 4.5. V tabeli tako vsaka vrstica predstavlja pridobljene rezultate za teren velikosti $1km \times 1km$, stolpec **Teren** pa pove za kakšen tip terena gre oz. kakšne terenske značilnosti prevladujejo.

4.4 Analiza pridobljenih rezultatov in primerjava z izhodiščno metodo

V tem poglavju bomo predstavili in analizirali podatke prikazane v Tabelah 4.2 in 4.3, ter si ogledali njihovo primerjavo z rezultati, pridobljenimi z referenčno metodo. Predstavili jih bomo v štirih delih; v vsakem od njih pa si bomo pogledali uspešnost in zanesljivost našega algoritma na enem od testnih terenov.

Standardni teren

Vrednosti za standardni teren so predstavljene v prvih vrsticah Tabel 4.2 in 4.3. Standardni tipi terena predstavlja najbolj pogost itip terena, na katerega smo naleteli pri iskanju jarkov; gre za območja, kjer so jarki relativno dobro ohranjeni (globoki, ostro definirani robovi), ne vsebujejo večjih



Slika 4.5: Rezultat našega algoritma za standardni tip terena

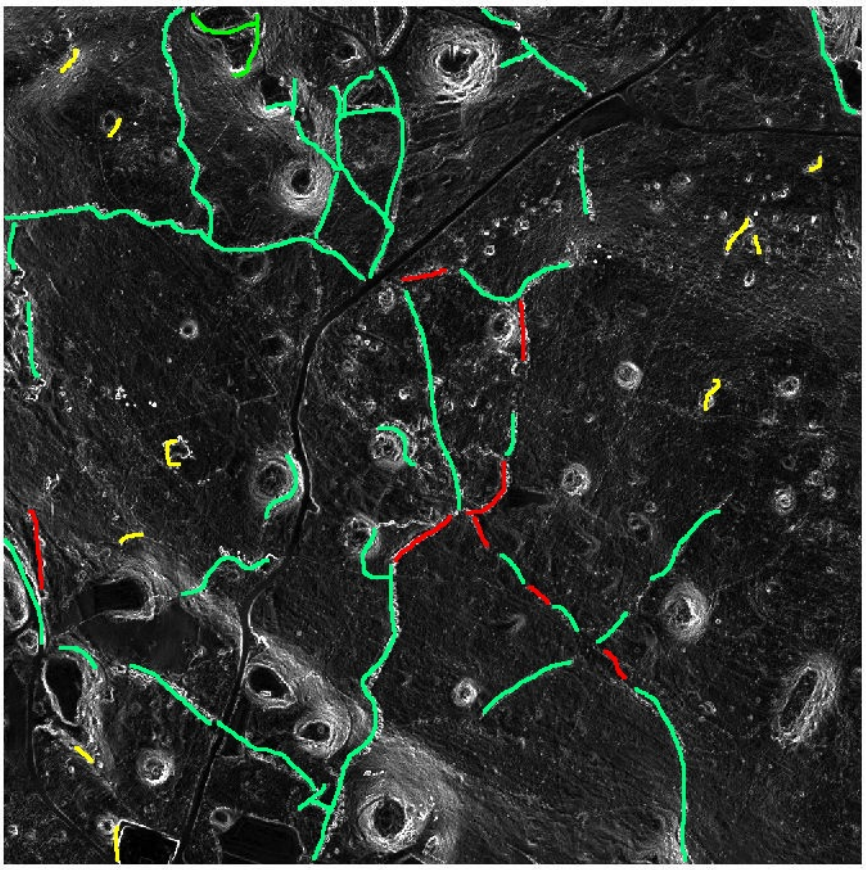
.

mikro in makro-topoloških anomalij, prav tako pa ne vsebujejo človeških posegov v naravo. Pri človeških posegih so ceste izvzete, saj le-te na delovanje našega algoritma ne vplivajo. Naš algoritem, ki smo ga pognali nad celotnim območjem, je tako dosegel naslednje rezultate: uspešno je prepoznal 79,45% vseh na roko vrisanih jarkov, hkrati pa je 72,72% vseh povezanih območji, ki jih je algoritem vrnil, predstavljalo jarke. S tem smo večinoma dosegli zastavljene norme, definirane v Poglavju 4.2, kjer smo definirali meje uspešnosti ($\Theta 1$) in zanesljivosti ($\Theta 2$):

$$\Theta_1 \geq 0.8, \tag{4.14}$$

$$\Theta_2 \geq 0.75. \tag{4.15}$$

V stolpcu **Interval vrednosti** je nato predstavljen še okvirni razpon vrednosti, katere smo dosegli, če smo algoritem poganjali na podobmočjih celotnega območja, kjer pa se pozna večji vpliv šuma in ostalih mikro-topoloških



Slika 4.6: Z zeleno so označeni jarki, ki jih je naš algoritem pravilno označili, z rdečo so označeni jarki, ki jih naš algoritem ni označil, z rumeno pa so označena območja, katera je naš algoritem označil za jarke, a to niso.

.

značilnosti na rezultat.

V zadnjem stolpcu pa so predstavljeni rezultati, katere je na celotnem območju dosegla metoda z uporabo Sobelovih filtrov. Kot bomo videli tudi v nadaljevanju, je v uspešnosti referenčna metoda vedno vračala boljše rezultate. Za standardni teren tako uspešno prepozna 99,65% jarkov, vendar je zanesljivost njena šibka točka. Že na standardnem terenu pade njena zanesljivost na 42,04% , kar je prav tista težava, zaradi katere smo večjo pomembnost v izde-

lavi našega algoritma namenili zanesljivosti. To v praksi pomeni, da le vsako drugo območje, ki ga referenčna metoda vrne, označuje jarek, ostala območja pa predstavljajo šum. Kljub prej naštetemu pa ima referenčna metoda vseeno potencial in bo uporabljena v nadaljnem razvoju našega algoritma.

Problematični teren - vrtače

Vrednosti za tak tip terena so predstavljene v drugih vrsticah Tabel 4.2 in 4.3. Gre za teren, kjer od mikro-topoloških značilnosti daleč najbolj prevladujejo vrtače. Pehotni jarki so, če je le možno, speljani po robovih vrtač oz. skozi vrtače, kar sicer naš algoritem upošteva, vendar uspešnost pade, če je vrtač preveč in se po možnosti še stikajo, kot se je to dogajalo na našem testnem območju.
Uspešnost našega algoritma je tu padla na 45,30% zanesljivost pa na 44,87%, kar je daleč od naših zastavljenih norm, vendar se bo na takih tipih terena rezultat dalo izboljšati v prihodnje, kar je opisano v Poglavju 5.2.
Referenčna metoda je tudi na tem tipu terena dosegla zelo visoko uspešnost prepoznave: 98,49% a je zanesljivost še dodatno padla na: 27,60%. Ponovno lahko opazimo, da si bomo z referenčno metodo lahko pomaglai v prihodnje, sama zase pa brez nadaljne obdelave ni dovolj zanesljiva, da bi jo lahko uporabili na takem tipu terena.

Problematični teren - človek

Vrednosti za tak tip terena so predstavljene v tretjih vrsticah Tabel 4.2 in 4.3. Tak tip terena vsebuje umetne posege v naravo in predstavlja hudo oviro za delovanje našega algoritma. Med umetne posege tu najbolj štejemo stopničasta pobočja, ki jih je ustvaril človek; npr. betonske odtočne kanale in ostale strukture, ki imajo podobne lastnosti kot pehotni jarki. Za testno območje smo tako izbrali kar najbolj problematično območje, kar smo jih našli, kjer poleg umetno zgrajenih struktur pehotnih jarkov skorajda ni. Uspešnost našega algoritma je bila kljub temu za problematične terene dokaj visoka 63,58% , močno pa je padla zanesljivost: 5,75%. Referenčna metoda

je ponovno dosegla visoko upešnost: 97,72%, kljub temu pa je imela nižjo zanesljivost od naše metode, z: 4,09%. Tudi za ta tip terena imamo v načrtu izboljšave, vendar se zavedamo, da bo to najverjetneje največja ovira, ki jo bomo morali še premagati.

Problematični teren - erozija

Vrednosti za tak tip terena, so predstavljene v četrtih vrsticah Tabel 4.2 in 4.3. Gre za iskanje jarkov, ki so bili izpostavljeni eroziji ter ostalim naravnim vplivom, zaradi česar so veliko slabše definirani in jih zato naš algoritem obravnava kot šum. Če na terenu prevladuje tak tip jarkov, potem uspešnost in zanesljivost našega algoritma strmo padeta. Na našem testnem območju je tako uspešnost padla na 18,48% in zanesljivost na 21,56%, referenčna metoda pa je ponovno dosegla zelo visoko stopnjo uspešnosti: 97,72% , a hkrati ponovno tudi slabšo stopnjo zanesljivosti: 15,50%. Tu je potrbno poudariti, da so taka območja redka. Največkrat se slabo definiran jarek nahaja v kombinaciji z dobro definiranimi pehotnimi jarki, na podlagi česar ga bomo sposobni identificirati v nadaljnem razvoju našega algoritma.

4.5 Kritični primeri

Spodaj so predstavljeni tereni oz. primeri terenov, na katerih je imel algoritem največ težav ter ideje, kako bi se lahko lotili reševanja le-teh. Več o idejah, ki jih bomo implementirali v prihodnje, pa je napisano v Poglavju 5.2.

Vrtače in druge mikro-topološke značilnosti

Gre za teren na katerem prevladujejo vrtače in ostale mikro-topološke značilnosti, ki niso jarki. Ker gre pri algoritmu za avtomatsko določanje pragovne vrednosti za absolutni algoritem - prag določi na podlagi vrednosti na celotni sliki, pride do nenatančnih odzivov. Prav tako na takem terenu pri vseh

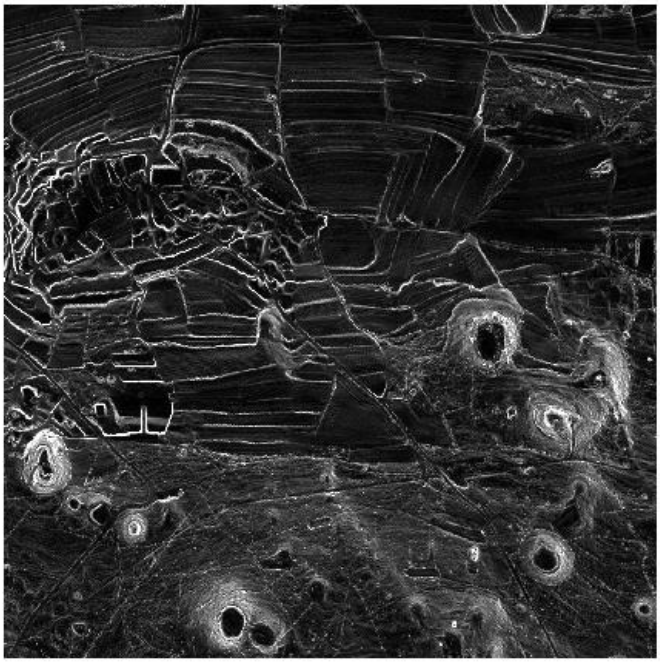
uporabljenih algoritmih dobimo poleg jarkov množico drugih pozitivnih odzivov, ki niso jarki, kar povzroča težave pri izločanju nepravilnih območji in posledično botruje slabšim rezultatom na takem tipu terena.

Človeški posegi v naravo

Še bolj so za algoritem moteči človeški posegi v naravo, še posebej taki, ki so na podlagi višinskih slik podobnih jarkom - imajo zelo jasne robove in podobno višinsko razliko. Algoritem praktično išče nepravilnosti v naravnem okolju, kar jarki so, vendar bo zato pozitivno označil vse umetno ustvarjene strukture, ki so jarkom podobne. Na kraškem terenu so tako najbolj moteča umetna stopničasta pobočja, namenjena vinogradom, težave pa povzročajo tudi npr. betonski odtočni kanali, kamniti zidovi med parcelami, itd. Testno območje, na katerem prevladujejo umetno ustvarjene stukture je prikazano na Sliki 4.7.

Posledice erozije in drugih naravnih pojavov

Tretji pojav, ki botruje slabšim rezultatom, pa je erozija oz. kar narava sama. Pehotni jarki, ki so na izpostavljenih območjih, kjer čutijo vpliv dežja, vetra. in niso vzdrževani, se s časom začnejo zapolnjevati. Tako so pri vizualnem pregledu terena sicer vidni, a niso več dobvolj jasno definirani, da bi jih algoritem zaznal. Če bi namreč prilagodil parametre tako, da bi zaznali tudi tako slabo definirane jarke, potem bi bil algoritem preveč občutljiv na vse ostale anomalije na terenu.



Slika 4.7: Testno območje, na katerem prevladujejo umetno ustvarjene strukture.

Poglavje 5

Zaključek

Algoritem, ki smo ga izdelali za diplomsko nalogo, se je glede na zastavljene kriterije odrezal dokaj solidno. Vsekakor pa smo pri izdelavi prišli do mnogih idej za izboljšave, ki še čakajo na implementacijo. V tem poglavju bomo predstavil ideje, kako se lotiti v prejšnjem poglavju predstavljenih kritičnih primerov, izboljšave, na katere smo naleteli, in metode, s katerimi smo se ukvarjali, a jih v končni različici nismo uporabili.

5.1 Ugotovitve

Kot smo opisali v Poglavju 4.5, smo pri izdelavi algoritma naleteli na problematična območja, na katerih je učinkovitost algoritma padla. Zaradi obširnosti naloge nam ni uspelo implementirati dodatnih algoritmov za posamezna območja. Ker se nameravamo s to tematiko ukvarjati tudi v nadaljnje, smo o rešitvi različnih problemov razmišljali in prišli do določenih idej, ki jih želimo implementirati v prihodnosti.

Najbolj pereč problem tako predstavljaja izjemna razgibanost terena. Imeli smo podatke za Slovensko primorje ter Posočje, kjer ne samo, da je poleg jarkov tudi množiica vrtač, ki se z jarki prepleta, temveč prevladujejo tudi ostale mikro in makro-topološke značilnosti, ter človeški posegi v naravo. Tu bi v poštev prišlo strojno učenje z algoritmom SVM [10, 19], za katerega

imamo ogrodje pripravljeno, vendar se je izkazalo, da je priprava učnih nizov preveč časovno zahtevna. S pravilnim učenjem bi se lahko prepoznava jarkov na takih terenih močno izboljšala. V našem primeru smo tako zaenkrat poskusili samo z učenjem na majhnih nizih, kjer so bili rezultati v najboljšem primeru primerljivi z značilnicamii, ki smo jih uporabili.
Naslednja ovira so bili človeški posegi v naravo in različni umetno ustvarjeni objekti. Ti so zaenkrat predstavljali problem, saj so glede na značilnice, s katerimi smo operirali, lahko izjemno podobni jarkom samim, vendar imajo jarki zanimivo lastnost, ki je zaenkrat še nismo uporabil pri klasifikaciji, a bo najverjetneje ključna pri ločevanju jarkov od ostalih umetno grajenih objektov. Jarki so namreč cik-cakasti, najpogosteje z 90 stopinjskimi zavoji vsakih 10-25 metrov, kar ni dobro poznano dejstvo. To smo opazili šele, ko smo vizualizirali podatke, ki jih je priskrbel sistem Lidar. V praksi to pomeni, da povezana območja, ki predstavljajo jarke ob tanjšanju, postanejo razvejana, kar je lepo vidno na praktično vseh rezultatih, ki jih algoritem vrne. Zaenkrat rezultati služijo za vizualno prepoznavo jarkov, tako da je dovolj, da se na oko lahko vidi, da ima rezultat značilno razvejano obliko. Z nekaj dela bomo to v prihodnosti prav tako implementirali v algoritem, saj bo na podlagi tega možno izračunati tudi verjetnosti, da neko območje v rezultatu pripada jarku, ter kako izrazit je jarek; oboje prav na podlagi razvejanosti.
Problematični so tudi jarki, ki so bili izpostavljeni vremenskim pojavom, zaradi česar niso več tako globoki in lepo definirani. Take jarke algoritem zavrne, saj bi bil preveč občutljiv na šume in ostale mikro-topološke značilnosti, če bi za pozitivna označil tudi tako neizrazita območja. Prepoznava takih jarkov pa na srečo pade tudi okrilje Poglavja 5.2, saj je eden od ciljev v prihodnosti tudi ta, da se uporabi rezultat algoritma, na podlagi katerega se pozitivnim območjem prilagodi premica, nato pa se poskuša te premice povezati v omrežja, na podlagi verjetnosti. Če bi bile tako dve ali več premic, ki označujejo jarke, dovolj blizu in bi zadostovale določenim pogojem, potem bi se take premice lahko povezalo, povezovalna premica pa bi vsebovala dodatni parameter verjetnosti. Za zelo izraziti premici, ki sta blizu, bo tako

povezovalna premica imela visoko verjetnost, da tudi ona označuje jarek. Na ta način bodo označeni tudi manj izraziti jarki, saj se bo dalo izkoristiti dejstvo, da so sistemi jarkov močno povezani. Na ta način so jih namreč gradili, tako da popolnoma izoliranih jarkov skorajda ni, oz. ločene skupine jarkov predstavljajo frontne črte.

5.2 Uporaba in nadaljnje delo

Algoritem smo izdelovali v sodelovanju z Inštitutom za vodarstvo. Osnovna ideja je, da se algoritem v prihodnosti požene čez celotno območje, na katerem se predvideva da se nahajajo jarki; zaenkrat je to slovensko Posočje in Primorje, pripravljene pa imamo tudi podatke za Italijo. Nato bi se rezultati vizualizirali čez teren tako, da bi lahko na njihovi podlagi vizualno pregledali, kateri jarki so zaenkrat vrisani v različnih arhivih in kateri niso.

Nato bodo sledili še dodatni koraki izboljšanja algoritma. Glede na pridobljene rezulate, se bomo odločili ali bomo implementirali korake strojnega učenja, vsekakor pa sledi implementacija idej, ki sem jih omenil v prejšnjem poglavju. Tem rezultatom bomo nato priredil premice, ter jih z določenimi verjetnostmi povezali v omrežja, tako da bo v končni fazi rezulat našega algoritma vektrorska mapa z označenimi pehotnimi jarki, kjer bo mogoče označene jarke sortirati in prikazovati glede na verjetnost, da tudi v realnosti označujejo jarke. Primarno prepoznani jarki bodo tako imeli 100 odstotno verjetnost, nato pa postopoma manj, glede na verjetnost, ki bo izračunana med povezovanjem.

Literatura

[1] Gauss-kruger image. https://upload.wikimedia.org/wikipedia/fi/b/b2/GaussKruger_Konstruktio.jpg.

[2] Gauss-kruger image. https://upload.wikimedia.org/wikipedia/commons/c/c2/Lbp_neighbors.svg.

[3] Lidar components image. http://www.ugr.es/~geomet/doc/introduction_lidar.pdf.

[4] *Understanding GPS: Principles and Applications Second Edition*. Artech House, Inc., 2006.

[5] *Trench Warfare 1850–1950*. Pen and Sword, 2010.

[6] *Computer Vision Using Local Binary Patterns*. Springer Science & Business Media, 2011.

[7] Timo Ahonen, Student Member, Abdenour Hadid, Matti Pietikäinen, and Senior Member. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2037–2041, 2006.

[8] E.P. Baltsavias. Airborne laser scanning: existing systems and firms and other resources. *ISPRS Journal of Photogrammetry & Remote Sensing*, pages 164–198, 1999.

[9] Roger G. Barry, Martin W. Miles, Richard C. Cianflone, G. Scharfen, and Russell C. Schnell. Characteristics of arctic sea ice from remote-sensing data and their relationship to atmosferic processes. *Annals of Glaciology*, pages 9–15, 1989.

[10] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, pages 121–167, 1998.

[11] Chi Ho Chan. *Multi-scale Local Binary Pattern Histogram for Face Recognition*. PhD thesis, University of Surrey, 2008.

[12] Rod Deakin, Max N. Hunter, and Charles F. F. Karney. Presented in Samia Belbachir (ed.) Proceedings of the 25th International Cartographic Conference (ICC2011), Paris, France, 3-8 July 2011, pp. 1-22.

[13] Zack G.W, Rogers W.E., and Latt S.A. Automatic measurement of sister chromatid exchange frequency. *J Histochem Cytochem*, pages 741–53, 1977.

[14] Mei-Po Kwan and Daniel M. Ransberger. Lidar assisted emergency response: Detection of transport network obstructions caused by major disasters. *Computers, Environment and Urban Systems*, page 179–188, 2009.

[15] Ivan Laptev, Helmut Mayer, Tony Lindeberg, Wolfgang Eckstein, Carsten Steger, and Albert Baumgartner. Automatic extraction of roads from aerial images based on scale-space and snakes. *Machine Vision and Applications*, pages 23–31, 2000.

[16] Yangming Li and Edwin B. Olson. Extracting general-purpose features from lidar data. *Robotics and Automation (ICRA)*, pages 1388 – 1393, 2010.

[17] Ku Chin Lin. Fast image thresholding by finding the zero(s) of the first derivative of between-class variance. *Machine Vision and Application*, pages 254–262, 2003.

[18] Myntti Matleena. Target identification with hyperspectral lidar. Master's thesis, Aalto University, 2015.

[19] David Meyer, Friedrich Leischa, and Kurt Hornikb. The support vector machine under test. *Neurocomputing*, 55:169–186, 2003.

[20] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. *ECCV'10 Proceedings of the 11th European conference on Computer vision: Part VI*, pages 210–223, 2010.

[21] RK Newsom. Doppler lidar (dl) handbook, 2012.

[22] Timo Ojala. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence*, pages 971 – 987, 2002.

[23] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, pages 51–59, 1996.

[24] Alexander Prokop, Peter Schön, Florian Singer, Gaëtan Pulfer, Mohamet Naaim, and Emmanuel Thibert. Determining avalanche modelling input parameters using terrestrial laser scanning technology. *International Snow Science Workshop Grenoble*, pages 770–774, 2013.

[25] Maria Belmonte Rivas, James Maslanik, J. G. Sonntag, and Penina Axelrad. Sea ice roughness from airborne lidar profiles. *Geoscience and Remote Sensing*, pages 3032 – 3037, 2006.

[26] R.D. Sanchez and J. Mullins. Integrated gps-aided inertial lidar and optical imaging systems for aerial mapping. Technical report, U.S. Geological Survey, 2004.

[27] Richard D. Sanchez and Kenneth W. Hudnut. Gps-aided inertial technology and navigation- based photogrammetry for aerial mapping the san andreas fault system. Technical report, U.S. Geological Survey, 2004.

[28] Russell Schnell, Roger Graham Barry, M. W. Miles, E. L. Andreas, L. F. Radke, and C. A. Brock. Lidar detection of leads in arctic sea ice. *Nature*, pages 530–532, 1989.

[29] Amos Sironi, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection by learning a scale-space distance transform. Technical report, Computer Vision Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL) and Institute for Computer Graphics and Vision, Graz University of Technology.

[30] Amos Sironi, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection by learning a scale-space distance transform. *Computer Vision and Pattern Recognition*, pages 2697 – 2704, 2014.

[31] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. Presentation to Stanford Artificial Intelligence Project 01/1968.

[32] Engin Turetken, Fethallah Benmansour, Bjoern Andres, German Gonzalez, Christian Blum, Hanspeter Pfister, and Pascal Fua. Automated reconstruction of complex curvilinear structures using path classifiers and integer programming. Technical report, Computer Vision Laboratory CVLAB.

[33] Todd M. Wilson. Use of natural areas for monitoring long-term effects of climate change. Technical report, Pacific Northwest Research Station, 2014.

[34] Huang Y., Sun X., and Hu G. An automatic integrated approach for stained neuron detection in studying neuron migration. *Microsc Res Tech*, pages 109–118, 2010.

[35] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conference*, July 2014.